DIPLOMARBEIT

Using Penalized Logistic Regression Models for Predicting the Effects of Advertising Material

ausgeführt am Institut für
Wirtschaftsmathematik
der Technischen Universität Wien

unter der Anleitung von
O. Univ. Prof. Dipl.-Ing. Dr. techn. Manfred Deistler

durch
Stefan Grosswindhager
Thalerstrasse 20
4452 Ternberg

Wien, am 19. Oktober 2009

Contents

Abstract

Marketing problems commonly involve classification of customers into "buyer" versus "non-buyer" or "responders" versus "non-responders". These cases usually require binary classification models such as logistic regression. This thesis reviews promising biased estimation techniques applied to logistic regression for the purpose of profiling and classifying higher-dimensional marketing data. The main objective of the thesis is the comparison of the predictive performance of the individual classifiers, taking widely used quantitative and qualitative measures into account, including receiver operating characteristics (ROC) and expected cost curves.

Acknowledgements

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Manfred Deistler, who has supported me throughout my thesis with his patience and knowledge whilst allowing me the room to work in my own way.

I also want to offer my regards and blessings to Dr. Margarete Überwimmer, Dipl.-Ing. Kerstin Pöttinger and Dipl.-Ing. Oliver Fränzl for their steady support in all matters.

I should not forget to say thanks to my friends, my student colleagues and the stock market for succeeding - thank goodness - in preventing me from finishing this thesis too quickly.

Finally, I want to thank my parents for everything they ever did for me and for always being there for me.

Chapter 1

Introduction

1.1 Focus of Thesis

This thesis is centered on logistic regressions applied to higher-dimensional binary classification problems. There are many fields of study, such as marketing, where it is important to predict a binary response variable, or equivalently the probability of occurrence of an event, in terms of the values of a set of explicative variables related to it. Logistic regressions often have a very high number of predictor variables so that appropriate methods for the reduction of the dimension are necessary. For this purpose and the purpose of strategical feature selection principal component logistic regression (PCLR), general partial least squares (GPLS) and elastic net models, including lasso regression, are introduced. The performance of these methods is tested on real marketing data, where the objective is to estimate probabilities of order of each individual customer and correctly classify them into potential responders and non-responders. For final model evaluation multiple criteria are proposed, including error rates like deviance and misclassification rate, the area under the receiver operating characteristic curve (AUC) and the widely used Akaike information criterion (AIC).

1.2 Organization of Thesis

The thesis is divided into six chapters. Following this introductory part is *Chapter 2*, which presents background information about Data Warehouses in general and briefly discusses the architecture of Data Warehouses. At the end of this chapter, a section gives insight into Data Mining and its main tasks.

The first part of *Chapter 3* individually introduces biased estimation methods for dimension reduction in the linear setting. The second part of *Chapter 3* covers the logistic regression model as well as the maximum likelihood procedure for the estimation of its parameters. Furthermore, ap-

proaches for the application of biased estimation techniques to logistic regressions are outlined. *Chapter 4* elucidates the role of regression diagnostics as a tool for appropriate model selection.

Chapter 5 deals with performance tests on real marketing data and includes a summary of the results. A brief conclusion is drawn in *Chapter 6*. Following this concluding chapter are several appendices.

Appendix A collects important methods and particular algorithms used throughout the thesis. *Appendix B* contains additional simulation results and miscellaneous, which did not fit elsewhere. For the sake of completeness, the entire source code is placed into *Appendix C*.

Chapter 2

An Introduction to Data Warehouses and Data Mining

2.1 Data Warehouse Definition

'Building the Data Warehouse' [41], published by B. Inmon, provides, rather than giving a scientific definition, a step-by-step how-to guide on building a data warehouse. This book contains the most widely used definition [30]:

> *A data warehouse is a subject-oriented, integrated, nonvolatile, and time-variant collection of data in support of management's decisions.*

Subject Oriented Data: Subjects to a data warehouse are data elements that are used to summarize information by [58]. The scale and varieties of subjects, for instance in retail business - *products* are sold to *customers* at certain *times* in certain *amounts* and at certain *prices* - strengthen the position of using multidimensional modeling. Queries then aggregate measure values over ranges of dimension values to produce results such as the total sales per month and product type.

Integrated Data: Integration may be considered as the process of mapping dissimilar codes to a common base, ensuring consistent and standardized data element presentations. The main difficulty is the maintenance in a consistent fashion, due to the fact that usually data warehouses are not loaded from one source of data, but multiple operational databases and even external sources [58]. Figure 2.1 illustrates an integration path from the application-oriented operational environment to the data warehouse. In the past, however, application designers did not consider the eventuality that the data representations they were operating on, would ever have to be integrated with other data. For instance, 'male' and 'female' could be coded by

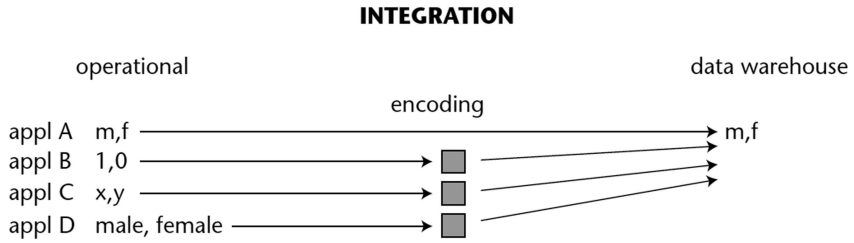


Figure 2.1: The issue of integration [41].

'M' and 'F' or '0' and '1'. Regardless of the method of coding in the operational environment, the primary issue is that warehouse encoding is done in a consistent way.

Time Variant Data: A data warehouse is time variant in the sense that it maintains both historical and current data. In contrast, operational environments only contain current-value data, or data whose accuracy is valid as of the moment of access [41]. For example, the current money on the customer's deposit is known by the bank at any time, whereas data warehouses are trying to store series of snapshots of the money on your deposit in daily, weekly, monthly or yearly time frames, leading to a sequence of historical data of events and activities. These historical information is vitally important for the decision making in the higher management, which are mainly based upon the understanding of trends and relationships between data, pointing out the practical relevance of statistical methods.

Non Volatile Data: The next characteristic of data warehouses is non-volatility, meaning that data is loaded into the warehouse in static format, hence changes, inserts or deletes are no longer performed unsystematically. In the case of subsequent changes, a new snapshot record is written, keeping the historical record of data in the data warehouse. In practice, updates are performed on a periodic basis, like weekly or monthly. Considering figure 2.2 in operational environments, as opposed to data warehouses, data records are regularly accessed and manipulated [41].

Besides these four primary principals, the challenging role of data warehouses is the support of *management decisions* and the supply with *strategic information*. Access to vital strategic information can substantially increase the quality of the decision-making process and can help in developing successful business strategies.

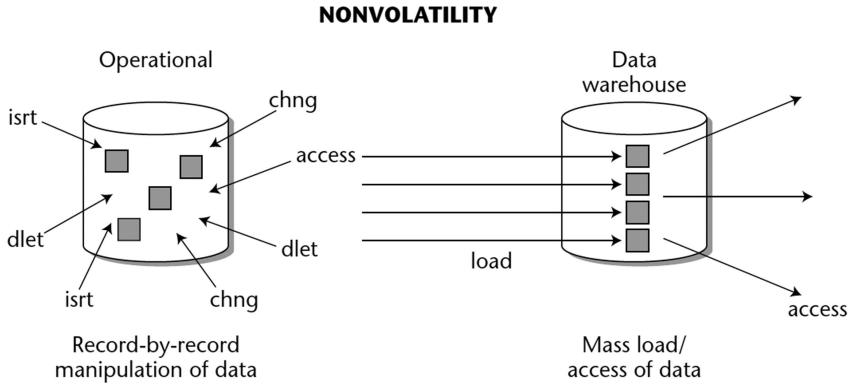


Figure 2.2: The issue of non-volatility [41].

2.2 Architecture of Data Warehouses

Architecture is the proper arrangement of the components.[1]

Data warehouses are typically built by a dozen of blocks of different hardware and software components, resulting in difficulties for the companies to arrange these building blocks benefit maximizing, but also given vast variation possibilities. Figure 2.3 illustrates the basic architecture, showing the *Source Data* component on the left, followed by the *Data Staging* and *Data Storage* component, which manage the data warehouse data. The final block *Information Delivery*, as the name implies, makes the information of the data available to potential users.

Source Data Component: Generally, source data components can be grouped into four categories, namely *Production, Internal, Archived and External Data* [55]. Production data arises out of numerous heterogeneous operational systems of the company. Heterogeneity in this context refers to the variations in the data formats (no standardization) and different hardware and software platforms, making it difficult for information queries to run across different operational systems. Hence, the significant and disturbing characteristic of production data is disparity. Business reports, customer profiles and company related documents are stored as internal data. Particularly the data privacy of customer profiles is a touchy issue, despite its importance for marketing consideration, such as making specific offerings to individual customers. Any historical information or snapshots of historical data, vital for time series analysis, is kept in the archive data. The time horizon of storage and the frequency of periodical data update depends on the

[1] Paulraj Ponniah [55].

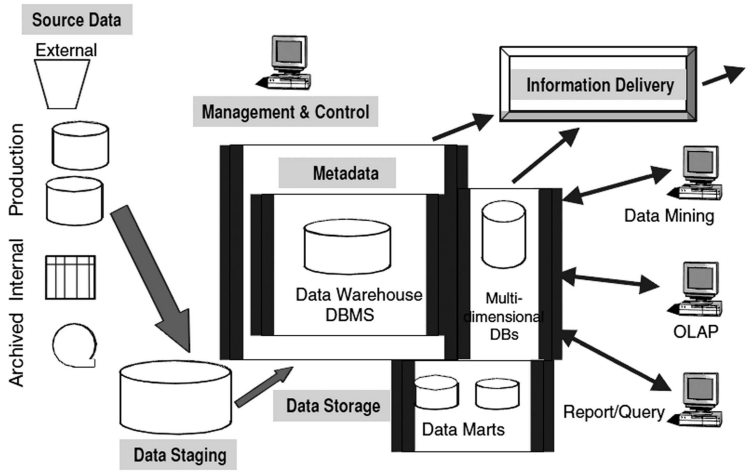


Figure 2.3: Data warehouse: building blocks or components [55].

requirements of the company. Production data and archived data, however, only provide a limited picture of the industry based on what the enterprise is doing or has done. In order to spot industry trends and compare performance against other organizations also external data is needed.

Data Staging Component: The data staging block deals with *Data Extraction, Data Transformation and Data Loading*. The former may become quite complex, due to, as already mentioned previously, numerous data sources and possible difficulties in employing an appropriate extraction technique for each data source. Even more challenging than data extraction seems to be the part of data transformation, which comprises cleaning and correction of misspelling of source data and standardization of data elements. For example, consider the problem of semantic standardization. Whether two or more terms from different source systems mean the same thing or a single term means many different things in different source systems, has to be resolved as synonym or homonym, respectively. Anyway, the goal of the data transformation function is to have a collection of integrated data at the end. Finally, the data loading function initially loads the data into the data warehouse storage and refreshes the data based on fixed (e.g. yearly, monthly, daily,...) cycles.

Data Storage Component: The data storage component is responsible for the data repositories of historical and current data. For the matter of user convenience many data warehouses also employ multidimensional database management systems, allowing data extraction from the data warehouse storage and summarizing the aggregated data in multidimensional databases.

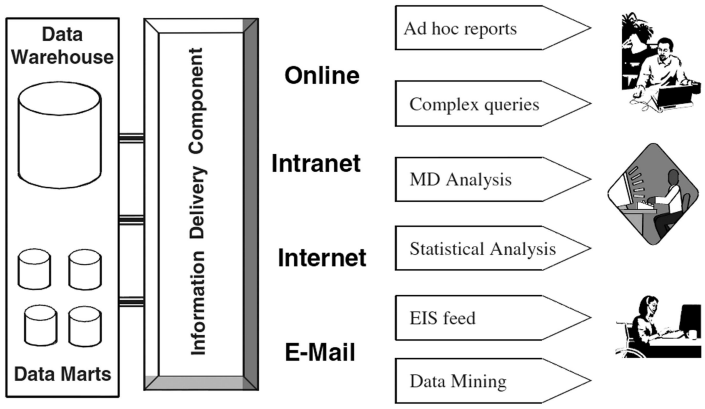


Figure 2.4: Information delivery component [55].

Information Delivery Component: The range of interest in information delivery is fairly comprehensive. It goes from ad hoc reports and complex queries to statistical analysis (fig. 2.4). First the progress in modern means of communication and information platforms made it possible, or rather more easier, to provide a wide community with essential data. Entering requests or queries online and receiving the results or reports online is not only convenient for the users, but also efficient in terms of work.

Metadata Component: Metadata in a data warehouse is similar to a data dictionary that keeps information about the logical data structures and about file addressing and indexing. A commonly used definition describes the metadata component as *the data about the data in the data warehouse* [55].

Management and Control Component: Sitting on top of all other components and coordinating the services and activities within the data warehouse, the management and control component has to interact with the metadata component to be able to perform these tasks successfully.

2.3 Data Mining in Data Warehouses

Data mining is defined as the process of discovering interesting knowledge and meaningful patterns from large amounts of data stored in data warehouses and other information repositories [31, 69]. Or in other words [32]:

> *Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.*

In the last years the amount of data in our lives increased rapidly, and there is still no end in sight. Whether we are surfing the World Wide Web or buying groceries at the supermarket, our behavior, all the decisions and choices are recorded and stored in explosively growing databases. Take the search engine *google.com* as an example. Malicious tongues even claim that *Google* knows more about you than your spouse. Nowadays, however, a growing gap between the *generation* of data and our *understanding* of it can be observed. Despite the fact that we are inundated with data in most fields there are not enough trained analysts available who are skilled at translating all of this data into knowledge, and thence up the taxonomy tree into wisdom [47].

Further we shall take a closer look at data mining in customer relationship management (CRM) [7], where data mining can be used to improve the company's ability to form beneficial learning relationships with its customers. Let us assume a company is able to "understand" each customer individually, then it will surely have an advantage over potential competitors, because the company will know which customers are worth investing money and effort to hold on to and which not. Consider, for instance, mail order companies using order histories to decide which (type of) customer should be included in future mailings. If a specific segment of customers only orders from the catalog at Christmas time, the overall response rate could be increased by not mailing to this segment the rest of the year. However, forming a profitable learning relationship with the customers is far from being an easy job. To succeed in CRM a company must be able to [7]:

- Notice and remember what its customers are doing and have done over time
- Learn from what has been remembered
- Act on what has been learned to make customers more profitable

The first bulletpoint is about storing details of the customers' orders such as type of product, order prices or the size of the order. Typically, as discussed previously, these information is standardized and formatted kept in large databases in data warehouses, so that each customer has its own historical record. These days information gathering is a whole business branch and, believe it or not, a quite profitable one. Supermarkets selling the data of customers' buying habits gained through the customers' usage of the inconspicuous discount cards, also called "loyalty" cards, earning good money. The learning issue is probably the most challenging part by the fact that finding patterns in the historical data is not only difficult by reasons of noisy historical data, missing data or inconsistent data, but also because of troubles occurring in the context of not knowing what is the best method or model to apply. Let us assume we built a successful model producing meaningful

scores[2]. The result allows to increase the response rate to mailings by targeting fewer customers and therefore save money, but also permits precisely targeted special offers to be mailed out to a specific segment of customers.

2.3.1 Tasks of Data Mining

In this section we briefly discuss the main tasks that data mining is usually called upon to accomplish, and also give algorithmic insight to well known data mining methods without going to deep into the underlying theory.

Classification

Classification is characterized as a two-step process, in which in the first step (training phase) a classifier is built describing a predetermined set of data classes (training set). It also goes by the name of *supervised learning*, since the classification algorithm builds the classifier by learning from the training set, in contrast to unsupervised learning, in which class labels and the number of classes are not known in advance. The second step is about building some model that can be applied to unclassified data in order to classify it [31]. Well known classification methods are amongst others: *Decision Tree Induction, Neural Networks, k-Nearest-Neighbor or Bayesian Classification.* For more details we refer the reader to [31]. Let us just sketch the basic idea of the latter.

Naive Bayesian Classification Assume that D is a training set of tuples $\mathbf{X} = (x_1, x_2, \cdots, x_n)$ representing a n-dimensional attribute vector, depicting n measurements made on the tuple from n database attributes, respectively, $A_1, A_2, \cdots, A_n$. Each tuple $\mathbf{X}$ is assumed to belong to a predefined class as determined by another database attribute called the class label attribute. For example, let $\mathbf{X} = (45, 35000€)$, where A_1 and A_2 are the attributes age and yearly income, respectively. Let the class label attribute be *buys car*. The associated class label for $\mathbf{X}$ might be *yes* or *no* (i.e., buys car = yes). Furthermore, supposing m classes $C_1, C_2, \cdots C_m$ and given $\mathbf{X}$, the naive Bayesian classifier will predict that $\mathbf{X}$ belongs to the class having the highest posterior probability, conditioned on $\mathbf{X}$, thus we have to maximize $\mathrm{P}(C_i|\mathbf{X})$. By Bayes' theorem $\mathrm{P}(C_i|\mathbf{X})$ is given through

$$\mathrm{P}(C_i|\mathbf{X}) = \frac{\mathrm{P}(\mathbf{X}|C_i)\mathrm{P}(C_i)}{\mathrm{P}(\mathbf{X})} \tag{2.1}$$

As $\mathrm{P}(\mathbf{X})$ is constant for all classes, only the term $\mathrm{P}(\mathbf{X}|C_i)\mathrm{P}(C_i)$ needs to be maximized. However, computing $\mathrm{P}(\mathbf{X}|C_i)$ in data sets with many attributes

[2] A score is a way of expressing the findings of a model in a single number. Scores can be used to sort a list of customers from most to least loyal or most to least likely to respond [7].

would be computationally expensive. Hence, the naive assumptions, which is where the name originates from, of class conditional independence is usually made: $\mathrm{P}(\mathbf{X}|C_i) = \prod_{k=1}^{n} \mathrm{P}(x_k|C_i)$.

Estimation and Prediction

By far the most widely used approach for estimation and prediction of unknown variables, the outcome to some input data, is regression analysis. Ranging from *linear regression*, where the continuous output is a linear function of the input variables, to *nonlinear* and *generalized linear models* like logistic regression having categorical outcome. The latter will be broadly discussed in the following chapters. As a real example consider a mail order company budgeted for a mailing of 100,000 pieces. One option would be to send the mailings to randomly chosen customers from the company's customers pool. A clever company, however, would estimate "propensity to order" scores for each customer, and would send the available mailings to candidates with the highest scores.

Affinity Grouping

Affinity grouping, also known as *Association rules*, may concisely be expressed as the job of finding which attributes go together, i.e., the task of association seeks to uncover rules for quantifying the relationship between different attributes [7, 47]. It is commonly used to analyze the purchasing patterns of customers, which are of great assistance in the fields of product placement, catalog design and cross-marketing.

Let us have a look how association rules are defined. Assume D is a data set of observation tuples described by n attributes, $A_1, A_2, \cdots, A_n$, and class label attribute $C = A_{class}$. An item p is an attribute-value pair of the form (A_i, v), i.e., A_i is the attribute taking the value v. Then association rules are of the form $P = (p_1 \wedge p_2 \wedge \cdots \wedge p_l) \Rightarrow C$, where the rule antecedent is a conjunction of items $p_1, p_2, \cdots, p_l$ $(l \leq n)$ associated with a class label C. The support s for association rule $P \Rightarrow C$ is the fraction of observations in the union of the antecedent and consequent, and can be stated as the following

$$s = \mathrm{P}(P \cap C). \tag{2.2}$$

The confidence c, as measure of the accuracy of the rule, is defined as

$$c = \mathrm{P}(C|P) = \frac{\mathrm{P}(P \cap C)}{\mathrm{P}(P)}. \tag{2.3}$$

Association rules are called "strong", if those meet or surpass certain minimum support and confidence criteria. A mail order company analyzing their order histories may find out that of the 100 customers ordered from the summer catalog 20 bought swimwear, and of those 20 who bought swimwear 5

bought suncream. Thus, a possible association rule could be "If buy swimwear, then buy suncream" with a *support* of $5/100 = 5\%$ and a *confidence* of $5/20 = 25\%$.

Clustering

Clustering is the process of grouping a set of physical or abstract objects into classes of *similar* objects [31]. It falls into the category of unsupervised learning, as contrary to the methods of classification, it does not rely on predefined classes and is often connected with *data segmentation*, since clustering partitions large heterogeneous data sets into a number of more homogeneous groups. Alternatively, this technique may also be applied for *outlier* (values that are "far" away from any cluster) *detection* or serves as a preprocessing step for some other form of data mining or modeling, such as the previously introduced technique of classification, which would then operate on the detected clusters and the selected attributes. Although there are multiple methods of clustering, most of them focus on the so called distance-biased cluster analysis. [31]. In the following the widely used k - means clustering method is presented.

The k-Means Method The k-means method partitions a set of n objects into k clusters, so that the resulting intracluster similarity is high, but the intercluster similarity is low. The similarity is measured in regard to the mean value of the objects in a cluster. In the first step, the algorithm randomly selects k objects, each of which initially represents a cluster mean. Then for each of the remaining objects an object is assigned to the cluster to which it is the most similar. After this the new mean for each cluster is computed and the whole process is iterated until some convergence criterion is met. Typically, the *square-error criterion* is used [31], which is defined as the summed squared distances from the objects to its center mean and can be written as

$$\mathrm{Err} = \sum_{i=1}^{k} \sum_{p \in C_i} |p - m_i|^2 \tag{2.4}$$

where p is the point in space representing a given object and m_i is the mean of cluster C_i.

In concluding this chapter, it is quite appropriate to quote D. T. Larose [47], who writes in the context of data mining that the latter is not so much magic but rather depends on the human factor:

> *Without skilled human supervision, blind use of data mining software will only provide you with the wrong answer to the wrong question applied to the wrong type of data. The wrong analysis is worse than no analysis, since it leads to policy recommendations that will probably turn out to be expensive failures.*

Chapter 3

Regression Methodology

Databases stored in data warehouses typically used in data mining may have millions of records and thousands of variables to choose from. However, the use of too many variables for modeling unnecessarily complicates the interpretation of the analysis, may lead to overfitting and may violate the *principle of parsimony*[1]. Furthermore, it is quite unlikely that there is any correlation structure among a huge set of variables, leading to instability in the solution space and incoherent results, such as in multiple regression, where a multicollinear set of variables can result in a regression that is significant overall, even when none of the individual variables are significant [48]. To overcome these problems a class of alternative methods, which settle into a category called *biased estimation methods*, has been proposed. This class includes, amongst others, ridge regression, the so-called lasso, partial least squares, and also approaches based on principal component analysis.

Variable selection methods like subset regression tackle the problem of too many predictors by only taking a subset of variables into the regression. Principal component is a technique based on explaining the set of correlated predictors by a reduced number of uncorrelated ones with maximum variance. As per definition, principal component does not take the relationship between response variable (outcome) and the predictors into account, in contrast to partial least squares, seeking directions that have high variance and have high correlation with the response. Ridge and lasso regression, on the other hand, incorporate a regularization method to the predictors via adding a L2 or L1 penalty term, respectively, to the residual sum of squares in order to be able to control the complexity of the model.

The overall quality of an estimate $\hat{\beta}$ of β is usually measured by the mean square error (MSE) and is calculated as the sum of the variance of the

[1]Principle of parsimony or principle of simplicity: One should always choose the simplest explanation of a phenomenon, the one that requires the fewest leaps of logic.

estimate and the squared bias.

$$\mathrm{MSE}(\hat{\beta}) = \mathrm{E}[(\hat{\beta} - \beta)^2] = \mathrm{Var}(\hat{\beta}) + \Big(\mathrm{Bias}(\hat{\beta}, \beta)\Big)^2 \tag{3.1}$$

According to the MSE quality criteria biased estimation methods are preferred over unbiased estimation methods, like ordinary least squares under certain conditions, in situations where the reduction in variance exceeds the increase in bias.

3.1 Biased Estimation Methods

3.1.1 Stepwise Regression

Forward stepwise regression [15, 24] starts off with the intercept and then sequentially adds into the model the predictor that most improves the fit. A partial F-value or AIC[2] score, for example, compared to a selected or default enter criterion value determines if the variable is allowed to enter the model or not. Then the equation is examined to see if any variable should be deleted. The basic structure of forward stepwise is summarized in the pseudo - code in appendix A.3. Variations of stepwise regression include *backward stepwise regression*, which starts with a larger model and sequentially removes variables that contribute least to the fit, and techniques based on *Efroymson's procedure* [19], which combines forward and backward steps. One of these so called hybrid (consider both forward and backward moves at each step and select the "best" of the two) stepwise-selection procedures is implemented in the R-package `stats` in the function `step`. This procedure uses the AIC criterion for weighting the choice, i.e., at each step an add or drop will be performed that minimizes the AIC score.

Stepwise regression comes under criticism mainly for the unstable behavior of the process, in fact that relatively small changes in the data might cause one variable to be selected instead of another after which subsequent choices may be completely different. Moreover, G. Edirisooriya [16] discusses the vulnerability of (ordinary) stepwise regression to specification, sampling and measurement errors, and concludes that stepwise regression may derive a theoretically and practically useless explanation of the underlying model.

In the following biased estimation methods are introduced as an alternative to stepwise regression and other unbiased regression techniques, hoping that the former may provide an improvement in terms of prediction performance.

3.1.2 Principal Component Analysis

Principal component analysis (PCA) is a common technique for finding patterns in data of high dimension in situations when the variables are very

[2] AIC is the abbreviation for Akaike Information Criterion. See subsection 4.2.1.

correlated. The basic idea is to build linear combinations of the original variables that capture most of the information, and using a subset of these transformed variables (principal components) for further modeling, thus reducing the dimensionality of the data. First introduced by Pearson (1901) and developed independently by Hotelling (1933) it success began, like other computationally intensive methods, with the advent of electronic computers.

Derivation of Principal Components

The computation of the principal components is straightforward, since it reduces to the solution of an eigenvalue-eigenvector problem for a positive-semidefinite symmetric matrix [42]. Consider a vector of random variables $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ and a coefficient vector[3] $\alpha_1 = (\alpha_{11}, \alpha_{12}, \ldots, \alpha_{1p})$ which maximizes $\mathrm{Var}(\alpha_1^{'}\mathbf{x}) = \alpha_1^{'}\mathbf{\Sigma}\alpha_1$, where $\mathbf{\Sigma}$ is the $p \times p$ variance/covariance matrix of $\mathbf{x}$. As it stands the optimization problem is unbounded. Therefore, for further derivations the normalization constraint $\alpha_1^{'}\alpha_1 = 1$ (the sum of squares of α is restricted to one) must be imposed. Moreover, employing the approach of Lagrangian multipliers for the optimization problem yields

$$\text{Max} \quad \alpha_1^{'}\mathbf{\Sigma}\alpha_1 - \lambda(\alpha_1^{'}\alpha_1 - 1), \tag{3.2}$$

where λ is a Lagrange multiplier. Differentiation with respect to α_1 gives

$$\mathbf{\Sigma}\alpha_1 - \lambda\alpha_1 = 0$$

or

$$(\mathbf{\Sigma} - \lambda\mathbf{I}_p)\alpha_1 = 0 \tag{3.3}$$

where $\mathbf{I}_p$ is the $p \times p$ identity matrix. Thus λ is an eigenvalue of $\mathbf{\Sigma}$ with corresponding eigenvector[4] α_1 [33]. The quantity to be maximized is now

$$\alpha_1^{'}\mathbf{\Sigma}\alpha_1 = \alpha_1^{'}\lambda\alpha_1 = \lambda\alpha_1^{'}\alpha_1 = \lambda. \tag{3.4}$$

Hence, for an optimum λ must be as large as possible, which is exactly when α_1 represents the eigenvector corresponding to the largest eigenvalue of Σ. Generally spoken, $\alpha_k^{'}\mathbf{x}$ is called the kth principal component (pc) of $\mathbf{x}$ with $\mathrm{Var}(\alpha_k^{'}\mathbf{x}) = \lambda_k$, where λ_k denotes the kth largest eigenvalue of $\mathbf{\Sigma}$, and α_k is the corresponding eigenvector. A proof for $k = 2$ is elucidated in [42, p. 5–6].

[3] These vectors are also called loadings and can be interpreted as eigenvectors multiplied by the square root of the corresponding eigenvalue.

[4] Note that eigenvectors corresponding to different eigenvalues are linearly independent.

Principal Components in Regression Analysis

Principal component regression tries to overcome the problem of multicollinearity, occurring when there are near linear functions of two or more of the predictor variables, as already briefly outlined in the introduction to this chapter, by simply using the principal components (pc's) in place of the original predictor variables. Due to the fact that the pc's are by definition uncorrelated exist no multicollinearities between them. If all of the pc's are included in the regression, the resulting model will be equivalent to that obtained by least squares, so the large variances caused by multicollinearities have not gone away [42]. Detailed strategies[5] for deciding which pc to delete from the regression equation are, amongst others, discussed in [42, Sec. 8.2] and [66]. Consider the standard regression model

$$\mathbf{y} = \mathbf{X}\beta + \epsilon \tag{3.5}$$

where $\mathbf{y}$ is a vector of n observations on the predictor variables, $\mathbf{X}$ is the $n \times p$ design matrix whose (i, j)th element is the value of the jth predictor variable for the ith observation, β is a vector of p regression coefficients and ϵ is a vector of white noise error terms. Further assume that the model has been mean centered[6] and the predictor variables have been standardized[7], so that $(\frac{1}{n-1})\mathbf{X}'\mathbf{X}$ is proportional to the (sample)[8] population correlation matrix of the predictor variables. The values (scores) of the pc's for each observation are given by

$$\mathbf{Z} = \mathbf{X}\mathbf{A} \tag{3.6}$$

where the *(i, k)*th element of $\mathbf{Z}$ is the value of the kth pc for the ith observation[9] and $\mathbf{A}$ is a $p \times p$ (also called loading) matrix whose columns are the unit-norm eigenvectors of $\mathbf{X}'\mathbf{X}$. Bearing in mind that $\mathbf{A}$ is an orthonormal matrix, equation (3.5) may be rewritten as follows

$$\begin{aligned} \mathbf{y} &= \mathbf{X}\mathbf{A}\mathbf{A}'\beta + \epsilon \\ &= \mathbf{Z}\mathbf{A}'\beta + \epsilon \\ &= \mathbf{Z}\gamma + \epsilon \end{aligned} \tag{3.7}$$

which simply replaces the predictor variables by their pc's in the regression model. Principal component regression can also be defined as a reduced

[5] For instance, choosing the first M pc's whose variances are less than some cut - off level l^*.

[6] Mean-centering is achieved by subtracting the mean of the variable vector from all the columns of $\mathbf{X}$.

[7] Standardization is accomplished by dividing each element of the mean centered $\mathbf{X}$ by the root sum of squares of that variable vector.

[8] One should always differentiate between population principal components and sample principal components [42, chap. 2/3].

[9] The matrix $\mathbf{Z}$ is orthogonal and referred to as the score matrix.

model of (3.7) with $m < p$

$$\mathbf{y} = \mathbf{Z}_m \gamma_m + \epsilon_m \tag{3.8}$$

where $\mathbf{Z}_m$ and γ_m are the corresponding subsets of the original elements and ϵ_m is the appropriate error term. Applying least squares to estimate γ in (3.7) and after this estimate β from equation

$$\widehat{\beta} = \mathbf{A}\widehat{\gamma} \tag{3.9}$$

is equivalent to estimate $\widehat{\beta}$ by applying least squares to (3.5). The main advantage of pc regression over ordinary least squares is in the case when multicollinearities are present. By deleting a subset of the pc's, especially those with small variances, much more stable estimates of $\widehat{\beta}$ can be obtained. Consider the following restatements of equation (3.9):

$$\begin{aligned} \widehat{\beta} &= \mathbf{A}\widehat{\gamma} \\ &= \mathbf{A}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{y} \\ &= \mathbf{A}\mathbf{L}^{-2}\mathbf{Z}'\mathbf{y} \\ &= \mathbf{A}\mathbf{L}^{-2}\mathbf{A}'\mathbf{X}'\mathbf{y} \\ &= \sum_{k=1}^{p} l_k^{-2} a_k a_k' \mathbf{X}'\mathbf{y} \end{aligned} \tag{3.10}$$

where l_k^2, the kth diagonal element of $\mathbf{L}^2$ ($\mathbf{L}$ is the diagonal matrix of the singular values), is the kth largest eigenvalue[10] of the matrix $(\mathbf{X}'\mathbf{X})^{-1}$ and a_k for $k = \{1, 2, \ldots, p\}$, the columns of A, are the unit-norm eigenvectors of $(\mathbf{X}'\mathbf{X})^{-1}$. Furthermore, making the usual assumption that $\mathrm{Cov}(\mathbf{y}) = \sigma^2\mathbf{I}_n$, the covariance of $\widehat{\beta}$ can be written as [42]

$$\begin{aligned} \sigma^2\mathbf{A}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{Z}'\mathbf{Z}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{A}' &= \sigma^2\mathbf{A}(\mathbf{Z}'\mathbf{Z})^{-1}\mathbf{A}' \\ &= \sigma^2\mathbf{A}\mathbf{L}^{-2}\mathbf{A}' \\ &= \sigma^2 \sum_{k=1}^{p} l_k^{-2} a_k a_k'. \end{aligned} \tag{3.11}$$

The last equation depicts that any predictor variable having moderate or large coefficients in any of the principal components, associated with small eigenvalues, will have large variance in the elements of $\widehat{\beta}$. To overcome this dilemma one could utterly delete terms in (3.10) that correspond to very small l_k^2, which may lead to the following estimator with $m < p$

$$\widetilde{\beta} = \sum_{k=1}^{m} l_k^{-2} a_k a_k' \mathbf{X}'\mathbf{y}. \tag{3.12}$$

[10]Note that the eigenvectors of $(\mathbf{X}'\mathbf{X})^{-1}$ are the same as those of $(\mathbf{X}'\mathbf{X})$, and that the eigenvalues of $(\mathbf{X}'\mathbf{X})^{-1}$ are the reciprocals of those of $(\mathbf{X}'\mathbf{X})$.

However, decreasing the variance of the estimator $\widehat{\beta}$ by omitting terms is achieved at the expense of introducing bias into the estimator $\widetilde{\beta}$. It follows from

$$\widetilde{\beta} = \widehat{\beta} - \sum_{k=m+1}^{p} l_k^{-2} a_k a_k^{'} \mathbf{X}^{'} \mathbf{y}$$

with $E[\widehat{\beta}] = \beta$ and $\mathbf{X}^{'}\mathbf{X} = \sum_{i=1}^{p} l_i^2 a_i a_i^{'}$[11], that

$$\begin{aligned} E(\widetilde{\beta}) &= E(\widehat{\beta}) - E\left[\sum_{k=m+1}^{p} l_k^{-2} a_k a_k^{'} \mathbf{X}^{'} \mathbf{y}\right] \\ &= \beta - \sum_{k=m+1}^{p} l_k^{-2} a_k a_k^{'} \mathbf{X}^{'} \mathbf{X} \beta \\ &= \beta - \sum_{k=m+1}^{p} a_k a_k^{'} \beta. \end{aligned} \tag{3.13}$$

The last term, in general, is non-zero, implying that $E(\widetilde{\beta}) \neq \beta$, thus we lost the unbiasedness property. Nevertheless, in situations where multicollinearity is a severe problem, the reduction in variance can outweigh the loss of unbiasedness of the estimator, as the often discussed issue of trade-off between bias and variance.

PCA by Singular Value Decomposition

Additional insights and a different way of interpretation of PCA is gained by showing mathematical relations to singular value decomposition (SVD). After some knowledge about PCA it suggests itself to use SVD as a widespread implemented and computationally efficient method for solving eigenvalue problems of square and symmetric matrices, and, moreover, for the derivation of the the principal components [42, 67]. Let $\mathbf{X}$ be a centered matrix of dimension $n \times p$ with rank[12] k. We can write the SVD of $\mathbf{X}$ as the following

$$\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^{'}. \tag{3.14}$$

Where $\mathbf{U}$, $\mathbf{V}$ are the $n \times p$, $p \times p$ orthonormal matrices, respectively, and $\mathbf{L}$ denotes, as earlier in this section, the $p \times p$ diagonal matrix with the high-to-low sorted singular values of $\mathbf{X}$. The columns of $\mathbf{U}$ are also called the *left singular vectors* and the rows of $\mathbf{V}'$ the *right singular vectors*. Furthermore, due to $rank(\mathbf{X}) = k$, $l_i = 0$ for $(k+1) \leq i \leq p$. One import result of SVD is that

$$\mathbf{X}^{(k)} = \sum_{i=1}^{k} \mathbf{u}_i l_i \mathbf{v}_i^{'} \tag{3.15}$$

[11] Is the outcome of the well-known spectral decomposition of $\mathbf{X}^{'}\mathbf{X}$ [42]. Proof can be found in any good book on functional analysis or linear algebra.

[12] The rank of a matrix is the number of linearly independent rows or columns.

minimizes the euclidean distance of the elements of $\mathbf{X}$ and $\mathbf{X}^{(k)}$, $||X-X^{(k)}||_2$, over all $n \times p$ matrices with rank k [42]. A feasible way to calculate the SVD is first to compute $\mathbf{V}'$ and $\mathbf{L}$ by diagonalizing $\mathbf{X}'\mathbf{X}$

$$\mathbf{X}'\mathbf{X} = \mathbf{V}\mathbf{L}^2\mathbf{V}' \tag{3.16}$$

and subsequently calculate $\mathbf{U}$ out of

$$\mathbf{U} = \mathbf{X}\mathbf{V}\mathbf{L}^{-1}. \tag{3.17}$$

Through centering of $\mathbf{X}$ is $\mathbf{X}'\mathbf{X}$ proportional to the covariance matrix. So, by equation (3.16), diagonalization of $\mathbf{X}'\mathbf{X}$ yields $\mathbf{V}'$, which also yields the principal components. In other words it can be stated that the right singular vectors are the same as the principal components of $\mathbf{X}$. Moreover, the eigenvalues of $\mathbf{X}'\mathbf{X}$ are equivalent to l_k^2, which are proportional to the variances of the principal components, and finally the matrix $\mathbf{UL}$ contains the principal component scores.

3.1.3 Partial Least Squares

Partial least squares (PLS) was developed by W. Hold (1966) and other researchers at the University Institute of Statistics in Uppsala in Sweden. Initially intended for social science problems [60] having scarce information, PLS is recently also gaining popularity in chemometrics [12] and other fields, where its ability, handling data sets with more predictor variables than observations, is of great avail.

In short, partial least squares is a technique that generalizes and combines features from principal component analysis and least squares regression. Stone and Brooks [64] introduced a general class of regression procedures called *continuum regression*, in which ordinary least squares and principal component regression occupy the opposite ends of the continuous spectrum, with partial least squares lying in between. The intrinsic difference between PCA and PLS is that the former tries to find (principal) components that provide the best explanation of the predictor variables in $\mathbf{X}$, whereas the dependent variable $\mathbf{Y}$ is not taken into account. Partial least squares, on the other hand, is seeking for components from $\mathbf{X}$ that are also relevant for $\mathbf{Y}$. More precisely, PLS regression searches for a set of components (called *latent vectors*[13]), which perform a simultaneous decomposition of $\mathbf{X}$ and $\mathbf{Y}$ with the constraint that these components explain as much as possible of the *covariance* between $\mathbf{X}$ and $\mathbf{Y}$ [1]. The definition and derivations below mainly follow those by [34,45] and [1,27], providing a theoretical and a more practical algorithmic insight into the structure of PLS regressions.

[13]PLS is also sometimes called "Projection to Latent Structures" [5].

Derivation of Partial Least Squares

Consider a bilinear decomposition, named as *outer relation*, of the centered $n \times p$ data matrix $\mathbf{X}$ and the centered $n \times m$ (multivariate) response matrix $\mathbf{Y}$ in the following form

$$\mathbf{X} = \mathbf{T}\mathbf{P}' + \mathbf{E} \tag{3.18}$$

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}' + \mathbf{F} \tag{3.19}$$

where, by analogy with PCA, $\mathbf{T}$, $\mathbf{U}$ are called the *score* matrices of size $n \times k$, collecting the k ($k \leq p$)[14] extracted score vectors also termed as *latent variables* or PLS components. Furthermore, $\mathbf{P}$ of size $p \times k$ and $\mathbf{Q}$ of size $m \times k$ are the orthogonal *loading* matrices (unlike to PCA, $\mathbf{P}$ and $\mathbf{Q}$ do not have to be orthonormal cp. equ. (3.6)), and finally the $n \times p$ matrix $\mathbf{E}$ and the $n \times m$ matrix $\mathbf{F}$ denote the matrices of residuals caused by dimension reduction. In the extreme case $k = p$ we would deduce as much components as given predictors, hence $\mathbf{E} = 0$. The same applies to $\mathbf{Y}$ and $\mathbf{F}$, respectively. One important question is how to choose the latent vectors. In theory any set of orthogonal vectors spanning the column space of $\mathbf{X}$ could be used to play the role of $\mathbf{T}$, although this choice may be suboptimal for prediction purposes. PLS, on the other hand, specifies $\mathbf{T}$ and $\mathbf{U}$ with additional conditions, and in its classical form it is based on the nonlinear iterative partial least squares (NIPALS) algorithm introduced by W. Hold (1975). Consider weight vectors $\mathbf{w}$, $\mathbf{c}$ and denote the corresponding score vectors by $\mathbf{t} = \mathbf{X}\mathbf{w}$ and $\mathbf{u} = \mathbf{Y}\mathbf{c}$, then the basic idea behind the PLS method may be stated as the following optimization problem

$$(\mathbf{w}, \mathbf{c}) = \underset{||\mathbf{w}||=||\mathbf{c}||=1}{\operatorname{argmax}} \left\{\operatorname{Cov}(\mathbf{X}\mathbf{w}, \mathbf{Y}\mathbf{c})\right\}^2 . \tag{3.20}$$

It can be proven that $\mathbf{w}$ corresponds to the first eigenvector, the one with the largest eigenvalue, of $\mathbf{X}'\mathbf{Y}\mathbf{Y}'\mathbf{X}$ [49], and similarly, eigenvalue problems for the extraction of $\mathbf{c}$, $\mathbf{t}$ or $\mathbf{u}$ estimates can be derived [1]. An iterative solution to (3.20) is given by the classical NIPALS algorithm 3.1, using instead of singular value decomposition, as it is done in the classical PCA, only a series of least squares regressions.

Partial Least Squares Regression

Based on the iterative process of the NIPALS algorithm slightly different PLS regression methods have been proposed. In the following we shall concentrate on the so called PLS2 method and try to gain insight into the algorithmic structure. For more detailed explanations one could confer [39] or [43]. In

[14] Without loss of generality, we assume that $\mathbf{X}$ and $\mathbf{Y}$ have full column rank.

Algorithm 3.1: NIPALS.

1: Initialize $\mathbf{u}$ with random numbers ▷ Univariate case: $\mathbf{u} = \mathbf{y}$
2: **for** until convergence of $\mathbf{t}$ **do**
3: $\quad \mathbf{w} = \mathbf{X}'\mathbf{u}/\mathbf{u}'\mathbf{u}$ ▷ estimate $\mathbf{X}$ weights
4: $\quad ||\mathbf{w}|| \to 1$ ▷ normalize $\mathbf{u}$
5: $\quad \mathbf{t} = \mathbf{X}\mathbf{w}$ ▷ estimate $\mathbf{X}$ factor scores
6: $\quad \mathbf{c} = \mathbf{Y}'\mathbf{t}/\mathbf{t}'\mathbf{t}$ ▷ estimate $\mathbf{Y}$ weight
7: $\quad ||\mathbf{c}|| \to 1$ ▷ normalize $\mathbf{c}$
8: $\quad \mathbf{u} = \mathbf{Y}\mathbf{c}$ ▷ estimate $\mathbf{Y}$ scores
9: **end for**

addition to the outer relation defined in equation (3.19), we assume there exists a linear *inner relation* between the score vectors $\mathbf{t}$ and $\mathbf{u}$ as follows

$$\mathbf{U} = \mathbf{T}\mathbf{D} + \mathbf{H} \tag{3.21}$$

where $\mathbf{D}$ is a $k \times k$ diagonal matrix and $\mathbf{H}$ denotes the matrix of residuals with dimension $n \times k$. These two relations allow now to express the final model with a *mixed relation*

$$\mathbf{Y} = \mathbf{T}\mathbf{D}\mathbf{Q}' + \mathbf{F}^* \tag{3.22}$$

where again $\mathbf{F}^* = \mathbf{H}\mathbf{Q}^{'} + \mathbf{T}$ denotes the matrix of residuals. Our objective is to obtain estimates for the unknown matrices $\mathbf{T}, \mathbf{D}, \mathbf{U}, \mathbf{Q}$ and $\mathbf{P}$. Let $\mathbf{E_0}$ and $\mathbf{F_0^*}$, the initial matrices, be the centered and scaled matrices of $\mathbf{X}$ and $\mathbf{Y}$, respectively. First of all, at each step h, $h = 1 \dots k$, the vectors $\mathbf{t}_h$ and $\mathbf{u}_h$ are computed according to the NIPALS algorithm 3.1[15]. Then the factor loadings for $\mathbf{X}$ and $\mathbf{Y}$ are determined by the following regressions

$$\mathbf{p}_h = \mathbf{E}_{h-1}^{'}\mathbf{t}_h/\mathbf{t}_h^{'}\mathbf{t}_h$$
$$\mathbf{q}_h = \mathbf{F}_{h-1}^{*'}\mathbf{u}_h/\mathbf{u}_h^{'}\mathbf{u}_h$$

Afterwards

$$d_h = \mathbf{t}_h^{'}\mathbf{u}_h/\mathbf{t}_h^{'}\mathbf{t}_h$$

is calculated, which represents the hth diagonal element of $\mathbf{D}$. In the next step the matrices are updated ("deflated") by subtracting the effects of $\mathbf{t}$ from both $\mathbf{E_{h-1}}$ and $\mathbf{F_{h-1}^*}$ as below

$$\mathbf{E}_h = \mathbf{E}_{h-1} - \mathbf{t}_h\mathbf{p}_h^{'}$$
$$\mathbf{F}_h^* = \mathbf{F}_{h-1}^* - \mathbf{t}_h d_h \mathbf{q}_h^{'}.$$

[15] In the algorithm the matrices $\mathbf{X}$ and $\mathbf{Y}$ have to be replaced with $\mathbf{E}_{h-1}$ and $\mathbf{F}_{h-1}^*$.

Finally, the entire procedure is iterated until all the k ($h = 1 \dots k$) latent variables have been found.

We are now in the same position as in PCA. Instead of directly regressing $\mathbf{Y}$ on $\mathbf{X}$, we use $\mathbf{T}$ to calculate the regression coefficients and later convert the coefficient vectors back to the realm of the original variables. Consider equation (3.22) and (3.18), then an estimator for $\mathbf{Y}$ can be written as

$$\widehat{\mathbf{Y}} = \mathbf{X}\mathbf{B}_{\mathbf{PLS}} \quad \text{with} \quad \mathbf{B}_{\mathbf{PLS}} = \mathbf{P}^{'+}\mathbf{D}\mathbf{Q}^{'} \tag{3.23}$$

where $\mathbf{P}^{'+}$ is the Moore-Penrose pseudo-inverse of $\mathbf{P}^{'}$ [1]. Another representation of $\widehat{\mathbf{Y}}$ is given by taking the relationship $\mathbf{T} = \mathbf{X}\mathbf{W}(\mathbf{P}^{'}\mathbf{W})^{-\mathbf{1}}$ into account [49]. Plugging this into (3.22) yields

$$\widehat{\mathbf{Y}} = \mathbf{X}\mathbf{W}(\mathbf{P}^{'}\mathbf{W})^{-\mathbf{1}}\mathbf{D}\mathbf{Q}^{'} = \mathbf{X}^{'}\mathbf{U}(\mathbf{T}^{'}\mathbf{X}\mathbf{X}^{'}\mathbf{U})^{-\mathbf{1}}\mathbf{T}^{'}\mathbf{Y} \tag{3.24}$$

in which in the last equality the relations among $\mathbf{T}$, $\mathbf{U}$, $\mathbf{W}$, $\mathbf{P}$ and $\mathbf{D}\mathbf{Q}^{'}$ are taken care of.

Beside the advantages of PLS like, for instance, the ability to handle multicollinearity and robustness in the face of data noise, difficulties occur in interpreting the loadings of the latent variables and in testing the significance of the variables, since distributional properties of estimates are hardly known. Criticism is also coming due to the algorithmic structure of PLS compared to other more straightforward approaches. Overall one could state that PLS, as well as PCA, is favored as a predictive technique and not as an interpretive technique.

3.1.4 Ridge Regression

Ridge Regression [15, 24] is a variant of ordinary multiple linear regression whose goal is to circumvent the problem of instability arising, amongst others, from collinearity of the predictor variables. As already discussed, PCA and PLS build uncorrelated linear combinations of the predictor variables in order to handle the problem of multicollinearity. Ridge regression, on the contrary, works with the original variables and tries to minimize a penalized residual sum of squares

$$\widehat{\beta}^{ridge} = \underset{\beta}{argmin} \left\{ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\} \tag{3.25}$$

where $\lambda \geq 0$ is the complexity parameter, also referred to as regularization parameter, that controls the amount of shrinkage. Like ordinary least squares, ridge regression includes all predictor variables, but typically with smaller coefficients, depending upon the value of the complexity parameter. As it can be seen in (3.25), $\lambda = 0$ corresponds to the least squares regression. To further illustrate this, the very equation may be equivalently written

as objective function which minimizes the least squares sum and constraint which is the restricted sum of the squared coefficients. In practice, no penalty is applied to the intercept β_0, and variables are scaled to ensure invariance of the penalty term to the scale of the original data. It can be shown, assuming centered data and estimating β_0 by the mean of the response variable $\mathbf{y}$, that the solution to equation (3.25), in the following written in matrix form

$$RSS(\lambda) = (\mathbf{y} - \mathbf{X}\beta)^{'}(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^{'}\beta \tag{3.26}$$

is given by

$$\widehat{\beta}^{ridge} = (\mathbf{X}^{'}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{'}\mathbf{y} \tag{3.27}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix. Adding λ to the diagonal of $\mathbf{X}^{'}\mathbf{X}$ makes the problem non-singular, even if multicollinearity is present.

Further insight into the nature of ridge regressions is gained by employing singular value decomposition, as before in PCA. Let $\mathbf{X} = \mathbf{U}\mathbf{L}\mathbf{V}^{'}$ be the SVD of the centered $n \times p$ design matrix $\mathbf{X}$, where $\mathbf{U}$ and $\mathbf{V}$ are the $n \times p$ and $p \times p$ orthonormal matrices and $\mathbf{L}$ is a diagonal matrix of size $p \times p$ consisting of the singular values of $\mathbf{X}$ ordered by value. So we may rewrite the least squares equation

$$\mathbf{X}\widehat{\beta}^{ls} = \mathbf{X}(\mathbf{X}^{'}\mathbf{X})^{-1}\mathbf{X}^{'}\mathbf{y} \tag{3.28}$$

to

$$\mathbf{X}\widehat{\beta}^{ls} = \mathbf{U}\mathbf{U}^{'}\mathbf{y}. \tag{3.29}$$

Now the ridge solutions are

$$\begin{aligned} \widehat{\beta}^{ridge} &= (\mathbf{X}^{'}\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^{'}\mathbf{y} \\ &= \mathbf{U}\mathbf{L}(\mathbf{L}^2 + \lambda\mathbf{I})^{-1}\mathbf{L}\mathbf{U}^{'}\mathbf{y} \\ &= \sum_{j=1}^{p} \mathbf{u}_j \frac{\mathbf{l}_j^2}{\mathbf{l}_j^2 + \lambda} \mathbf{u}_j^{'}\mathbf{y} \end{aligned} \tag{3.30}$$

where the $\mathbf{u}_j$ are the columns of $\mathbf{U}$ and $\mathbf{l}_j^2$ are the diagonal entries of $\mathbf{L}^2$. According to equation (3.30), ridge regression shrinks the coordinates of $\mathbf{y}$ with respect to the orthonormal basis $\mathbf{U}$ by the factor $\mathbf{l}_j^2/(\mathbf{l}_j^2 + \lambda)$, which implies that greater amount of shrinkage is applied to basis vectors with smaller eigenvalues $\mathbf{l}_j^2$.

Methods for the Choice of λ

The fundamental idea of ridge regression is to choose a value of λ for which the reduction in total variance is not exceeded by the increase in bias. Since λ is directly related to the amount of bias introduced, it is desirable to select the smallest value of λ for which instability occurred due to the presence of multicollinearity is eliminated. To follow the notation and terminology used

in [36], we denote with $\widehat{\beta}^{ridge}(k)$ the class of ridge estimators indexed by a parameter k, where k simply plays the role of λ. According to equation (3.27), $\widehat{\beta}^{ridge}(k)$ can be written as

$$\widehat{\beta}^{ridge}(k) = (\mathbf{X}^{'}\mathbf{X} + k\mathbf{I})^{-1}\mathbf{X}^{'}\mathbf{y} = (\mathbf{X}^{'}\mathbf{X} + k\mathbf{I})^{-1}\mathbf{X}^{'}\mathbf{X}\widehat{\beta}^{\mathbf{ls}}. \tag{3.31}$$

Furthermore, employing equations (3.6) and (3.7) allow to restate ridge regression in terms of principal component analysis. First applying least squares to (3.7) yields

$$\widehat{\gamma} = (\mathbf{Z}^{'}\mathbf{Z})^{-1}\mathbf{Z}^{'}\mathbf{y} = (\mathbf{L^2})^{-\mathbf{1}}\mathbf{c} \tag{3.32}$$

as estimator for γ, where $\mathbf{L^2}$, as earlier in this chapter, denotes the diagonal matrix with the (positive) eigenvalues of $\mathbf{X'X}$ and $\mathbf{c}$ is a substitute for $\mathbf{Z}^{'}\mathbf{y}$. Likewise a general ridge procedure for $\widehat{\gamma}$ is defined from

$$\widehat{\gamma}^{ridge} = (\mathbf{Z}^{'}\mathbf{Z} + \mathbf{K})^{-1}\mathbf{Z}^{'}\mathbf{y} = (\mathbf{L^2} + \mathbf{K})^{-1}\mathbf{c} \tag{3.33}$$

where $\mathbf{K}$ is a diagonal matrix with non-negative elements. Noting that the relationship between the equations (3.31) and (3.33) can be written as (cp. equ. (3.9))

$$\widehat{\beta}^{ridge}(k) = \mathbf{A}\widehat{\gamma}^{ridge} \tag{3.34}$$

with $\mathbf{K} = k\mathbf{I}$ and where $\mathbf{A}$ consists of the unit-norm eigenvectors of $\mathbf{X'X}$ [36]. In order to determine the unknown matrix $\mathbf{K}$ and to find the optimal diagonal elements we want to minimize the mean square error (MSE) of $\widehat{\gamma}^{ridge}$ relative to γ

$$\mathrm{MSE}(\widehat{\gamma}^{ridge}, \gamma) = \mathrm{E}[(\widehat{\gamma}^{ridge} - \gamma)^{'}(\widehat{\gamma}^{ridge} - \gamma)] = \sum_{i=1}^{p} \mathrm{E}(\widehat{\gamma}_i^{ridge} - \gamma_i)^2. \tag{3.35}$$

Bearing in mind that if the error term is white noise $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$, the vector $\mathbf{c} = \mathbf{Z}^{'}\mathbf{y} = \mathbf{Z}^{'}\mathbf{Z}\gamma + \mathbf{Z}^{'}\epsilon = \mathbf{L^2}\gamma + \mathbf{Z}^{'}\epsilon$ in equation (3.33) is multivariate normal distributed with parameter $\mathbf{L^2}\gamma$ and $\sigma^{\mathbf{2}}\mathbf{L^2}$

$$\mathbf{c} \sim \mathcal{N}(\mathbf{L^2}\gamma, \sigma^{\mathbf{2}}\mathbf{L^2}).$$

Hence, rewriting the very equation (3.33) in scalar term and plugging this into (3.35) yields [36]

$$\begin{aligned}
\mathrm{MSE}(\widehat{\gamma}^{ridge}, \gamma) &= \sum_{i=1}^{p} \mathrm{E}\left[\frac{c_i}{l_i^2 + k_i} - \gamma_i\right]^2 \\
&= \sum_{i=1}^{p} \frac{\sigma^2 l_i^2}{(l_i^2 + k_i)^2}\ \mathrm{E}\left[\frac{c_i - l_i^2\gamma_i}{\sigma l_i} - \frac{\gamma_i k_i}{\sigma l_i}\right]^2 \\
&= \sum_{i=1}^{p} \frac{\sigma^2 l_i^2}{(l_i^2 + k_i)^2}\ \left[1 + \frac{\gamma_i^2 k_i^2}{\sigma^2 l_i^2}\right] \\
&= \sum_{i=1}^{p} \frac{\sigma^2 l_i^2 + \gamma_i^2 k_i^2}{(l_i^2 + k_i)^2}.
\end{aligned} \tag{3.36}$$

Let us differentiate the last equation with respect to the $k_i's$ to obtain the optimal value of k_i :

$$k_i = \frac{\sigma^2}{\gamma_i^2}. \tag{3.37}$$

Verifying the second order conditions shows that the values for k in equation (3.37) indeed constitute a minimum for equation (3.36). Conclusively, one may state that Hoerl and Kennard [36] have proven the existence of $k > 0$, so that the ridge estimator produces lower mean square error than the least squares estimator [15]. However, this result is relatively useless in practice, since σ and γ in (3.37) are unknown parameters the optimal values for k are also unknown. On account of this, they proposed more pragmatic methods for the choice of k. These methods include, amongst others, (detailed description can be found in [10, p. 271]): *Fixed Point Method*, *Iterative Method* and the so-called *Ridge Trace*[16]. Other common procedures for estimating the "ridge parameter" are methods based on cross validation introduced in the next subsection.

3.1.5 Lasso Regression

The least absolute shrinkage and selection operator introduced by Tibshirani [65] and abbreviated as lasso is a hybrid of variable selection and shrinkage estimator. This technique shrinks the coefficients of some of the variables not simply towards zero like ridge regression, but exactly to zero, giving an implicit form of variable selection. More formally, the method minimizes the residual sum of squares subject to a constraint on the sum of absolute values (L_1 penalty term) of the regression coefficients $\sum_{j=1}^{p} |\beta_j| \leq t$, which can be equivalently written as

$$\widehat{\beta}^{lasso} = \underset{\beta}{argmin} \left\{ \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}. \tag{3.38}$$

The similarity to ridge regression should be noticed, wherein the L_2 ridge penalty $\sum_{j=1}^{p} \beta_j^2$ in equation (3.25), is replaced by the L_1 lasso penalty term $\sum_{j=1}^{p} |\beta_j|$.
With an adaptation of PCA, as discussed in [42], even a relationship between PCA and lasso can be constructed. Consider the linear combinations $\mathbf{a}_k^{'}\mathbf{x}$ $k = 1, 2, \ldots, p$, of the p measured variables $\mathbf{x}$ that have maximum variance subject to $\mathbf{a}_k^{'}\mathbf{a}_k = 1$, a solution to equation (3.2), respectively. The adaptation of PCA is given through the additional constraint $\sum_{j=1}^{p} |a_{kj}| \leq t$, where a_{kj} is

[16]The ridge trace constitutes a graphical representation of the ridge estimator $\widehat{\beta}^{ridge}(k)$ as a function of k. By varying k, typically in the range between 0 to 1, and plotting the results of the estimation against k, an appropriate value can be found.

the jth element of the kth vector a_k and t is the "tuning" parameter. It can easily be seen that for $t \geq \sqrt{p}$ we would obtain the (ordinary) PCA solution. As per definition of lasso, making t sufficiently small in the constraint will cause some of the coefficients to be exactly zero. On the other hand, if t is chosen large enough, the lasso estimates equal the least squares solution, thus the lasso performs a kind of continuous subset selection [24]. Further insight into the shrinkage behavior can be gleaned from the orthonormal design case. Let $\mathbf{X}$ be a $n \times p$ design matrix and suppose that $\mathbf{X}'\mathbf{X} = \mathbf{I}$. Then it can easily be shown that solutions to equation (3.38) have the following form [23]

$$\hat{\beta}_j^{lasso} = \text{sgn}(\hat{\beta}_j^{ls})(|\hat{\beta}_j^{ls}| - \gamma)^+ \tag{3.39}$$

where γ is determined by the condition $\sum_{j=1}^{p} |\hat{\beta}_j^{lasso}| \leq t$ and $\hat{\beta}_j^{ls}$ is the ordinary least squares estimate.[17] Similarly, one can prove that in the orthogonal setting the solutions to the ridge regression (3.31) are

$$\hat{\beta}_j^{ridge} = \frac{\hat{\beta}_j^{ls}}{1+\gamma} \tag{3.40}$$

where γ depends on λ or t, respectively. Comparing (3.39) with (3.40) shows that ridge regression scales the least squares coefficients by a constant factor, whereas the lasso translates by a constant factor, truncating at zero. In the literature the latter is often termed as "soft thresholding" in the context of wavelet-based smoothing too [24].
In the first part of this subsection it was argued that lasso shrinks coefficients in the general (non-orthonormal) setting exactly to zero, in contrast to ridge regression. At least for the two dimensional case, figure 3.1 will provide some visual insight, whether this statement holds. The criterion $\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2$ equals the quadratic function

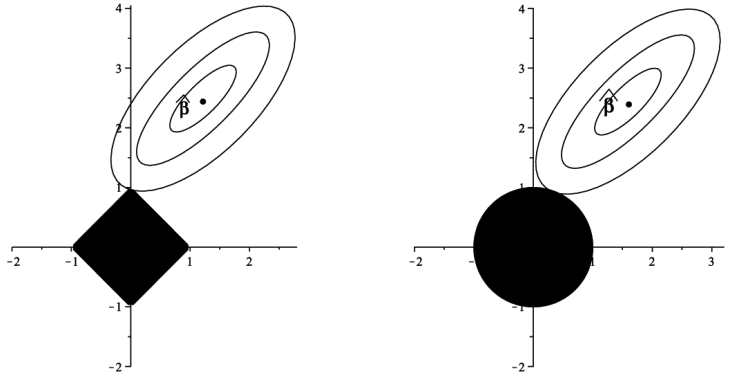
$$(\beta - \hat{\beta}^{ls})'\mathbf{X}'\mathbf{X}(\beta - \hat{\beta}^{ls}) \tag{3.41}$$

plus a constant error term, which may simply be derived from the normal equations. The elliptical contours of this function are shown by the full curves in figure 3.1, centered at the least squares estimates. The constraint regions are the rotated square (L_1 lasso penalty) in 3.1(a) and the circle (L_2 ridge penalty) in 3.1(b). The lasso solution is the first place that the contours touch the square, and this will sometimes happen at a corner, corresponding to a zero coefficient, whereas in ridge regression there are no corners for the contours to hit, thus zero solution will rarely result.

Estimation of the Regularization Parameter λ

The optimal regularization (shrinkage) parameter λ in ridge regression was derived by minimizing a squared error. Similarly, Tibshirani et al. [65] pro-

[17] $(|\hat{\beta}_j^{ls}| - \gamma)^+ = max(|\hat{\beta}_j^{ls}| - \gamma, 0)$.



(a) (b)

Figure 3.1: Estimation picture for the lasso (a) and the ridge (b) regression.

posed three methods that can be used for the purpose of minimizing some error criterion, and hence finding optimal lasso parameters λ and t, respectively: *Cross validation, general cross validation* and an *analytical unbiased estimate of risk*. In k-fold cross validation the data set is randomly divided into k subsets (subsamples) of roughly equal size. Then the model is fitted k times, each time leaving out one of the subsets from the available data, and for testing only the omitted subset is used to compute an error criterion. More detailed, let $\kappa : \{1, \ldots, n\} \to \{1, \ldots, k\}$ be an indexing function that indicates the partition to which the observation i is allocated by the randomization. Denote by $\hat{f}^{-k}(x)$ the fitted function calculated with the kth part of the data removed. Accordingly, the cross-validation estimate of the prediction error can be phrased as [24]

$$CV = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{f}^{-\kappa(i)}(x_i))^2. \tag{3.42}$$

Note that the prediction error of the model $\mathbf{y} = f(\mathbf{X}) + \epsilon$ with $\mathrm{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = \sigma_\epsilon^2$ is defined by

$$\mathrm{PE} = \mathrm{E}[\mathbf{y} - \hat{f}(\mathbf{X})]^2 = \mathrm{MSE} + \sigma_\epsilon^2. \tag{3.43}$$

Typically, the CV estimates are calculated over a grid of values of the normalized parameter $s = t / \sum \hat{\beta}^{ls}$ and the value $\hat{s}$ yielding the lowest estimated PE is selected.

The second method may be derived from a linear approximation[18] to the ridge estimate. Let us write the constraint $\sum |\beta_j| \leq t$ as $\sum \beta_j^2 / |\beta_j| \leq t$,

[18]The lasso estimate is a non-linear and non-differential function of the response values even for fixed values of t.

being equivalent to a Lagrangian penalty $\lambda \sum \beta_j^2/|\beta_j|$ with λ depending on t. Thus, substituting the new penalty term for $\lambda \sum \beta_j^2$ in the ridge equation (3.25) yields

$$\tilde{\beta} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{W}^-)^{-1}\mathbf{X}'\mathbf{y} \tag{3.44}$$

where $\mathbf{W} = diag(|\hat{\beta}_j^{lasso}|)$, $\mathbf{W}^-$ denotes the generalized inverse of $\mathbf{W}$ and λ is chosen so that $\sum |\tilde{\beta}_j| = t$. Then, according to Hastie [24] and also broadly discussed in [75], the *number of effective parameters* (effective degrees of freedom) in the constrained fit $\tilde{\beta}$, which is considered to be an informative measurement of the model complexity, may be approximated by

$$p(t) = \text{trace}(\mathbf{X}(\mathbf{X}'\mathbf{X} + \lambda \mathbf{W}^-)^{-1}\mathbf{X}'). \tag{3.45}$$

Let $\text{rss}(t)$ be the residual sum of squares for the constrained fit with constraint t. The generalized cross-validation statistic may be constructed as the following

$$GCV(t) = \frac{1}{n}\frac{\text{rss(t)}}{(1 - p(t)/n)^2}. \tag{3.46}$$

However, the last few lines may be easier to understand by having a look at the least squares setting. We can write the predictions of the standard model as

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} = \mathbf{S}\mathbf{y}. \tag{3.47}$$

The effective number of parameters, corresponding to equation 3.45, is defined as

$$\text{df}(\hat{\mathbf{y}}) = \text{trace}(\mathbf{S}). \tag{3.48}$$

Furthermore, one can show that $\sum_{i=1}^n \text{Cov}(\hat{y}_i, y_i) = \text{trace}(\mathbf{S})\sigma_\epsilon^2$, which motivates the more general definition

$$\text{df}(\hat{\mathbf{y}}) = \frac{\sum_{i=1}^n \text{Cov}(\hat{y}_i, y_i)}{\sigma_\epsilon^2}. \tag{3.49}$$

The third method is based on Stein's unbiased estimate of risk [62], in which equation (3.50) ($\mathbf{z} \sim \mathcal{N}(\mu, \mathbf{I})$; $\hat{\mu}$ is an estimate for μ written as $\hat{\mu} = \mathbf{z} + \mathbf{g}(\mathbf{z})$; $\mathbf{g}$ is an almost differentiable[19] function from $\mathcal{R}^p \to \mathcal{R}^p$) is basically applied to the lasso estimate in the orthonormal setting (equ. (3.39)). Then from minimization of $\text{E}\left[\hat{\beta}(\gamma) - \beta\right]^2$ one may derive an estimate for the shrinkage parameter t.

$$\text{E}_\mu||\hat{\mu} - \mu|| = p + \text{E}_\mu\left(||\mathbf{g}(\mathbf{z})||^2 + 2\nabla\mathbf{g}(\mathbf{z})\right). \tag{3.50}$$

[19] A function $h : \mathcal{R}^p \to \mathcal{R}$ is called almost differentiable if there exists a function $\Delta h : \mathcal{R}^p \to \mathcal{R}^p$, such that for all $\mathbf{z} \in \mathcal{R}^p$,

$$h(\mathbf{x} + \mathbf{z}) - h(\mathbf{x}) = \int_0^1 \mathbf{z} \cdot \Delta h(\mathbf{x} + t\mathbf{z})dt$$

for almost all $\mathbf{x} \in \mathcal{R}^p$. Moreover, a function $\mathbf{g} : \mathcal{R}^p \to \mathcal{R}^p$ is almost differentiable if all its coordinate functions are [62, def. 1].

As Tibshirani [65] argues, the last method, although assuming orthonormality, may also be tried to be used in general settings. Because with a standardized design matrix $\mathbf{X}$ one could state that the optimal value of the shrinkage parameter is roughly a function of the signal-to-noise ratio in the data, and relatively insensitive to the covariance of $\mathbf{X}$.

3.2 Logistic Regression Model

Logistic regressions (LOGIT) [11, 38], originated in the epidemiological research and now commonly employed in fields ranging from engineering to finance, are settled in the class of probability models. In contrast to ordinary linear regression, where the dependent variable has continuous nature, logistic regressions are dealing with qualitative response variables, specially the case of a binary (dichotomous) response. Let us examine, for instance, a mail order company more closely. Assume that the independent variable (or also called covariate) is household income, which is continuous, and the dependent variable (or response variable) is 'order from last mailing', being discrete. In our case the outcome Y is a scalar, which can take only two values, conventionally assigned the values 0 and 1. The event Y = 1 is designated as a success of the experiment and Y = 0 as a failure.

$Y_i = 1$ if household ordered from last mailing, and
$Y_i = 0$ otherwise.

Trying to find a linear relationship between the Y_i's and the X_i's, the household income, using least squares, may not be appropriate and is hardly justifiable. One reasonable way would be to construct the probability of $Y_i = 1$, not the value of Y_i itself, as suitable function of the covariate.

3.2.1 Model Specification

We consider a binary random variable y having Bernoulli distribution

$$y \sim \mathcal{B}(1, \pi(\mathbf{x})) \tag{3.51}$$

where $\mathbf{x} \in \mathcal{R}^p$ is a vector of p independent variables and $\pi : \mathcal{R}^p \to [0, 1]$ is the response probability function. The latter represents the conditional probability $\mathrm{P}(y = 1|\mathbf{x})$ of $y = 1$, given $\mathbf{x}$[20]. Let $e := y - \mathrm{E}(y|\mathbf{x}) = y - \pi(\mathbf{x})$ be an error term. Hence, we can rewrite our model as

$$y = \pi(\mathbf{x}) + e \tag{3.52}$$

with $\mathrm{E}(e) = 0$ and $\mathrm{Var}(e) = \pi(\mathbf{x})[1 - \pi(\mathbf{x})]$, which can be deduced from the characteristics of the Bernoulli distributed variable y. It is important to note that the variance of e depends on $\mathbf{x}$. Furthermore, the *logistic transformation* $G(\mathbf{z}) := \mathcal{R}^p \to [0, 1]$ defined by the following

$$G(\mathbf{z}) = \frac{\exp(\mathbf{z})}{1 + \exp(\mathbf{z})} \tag{3.53}$$

[20] If $\mathrm{P}(y = 1|\mathbf{x}) = \pi(\mathbf{x})$ then $\mathrm{P}(y = 0|\mathbf{x}) = 1 - \pi(\mathbf{x})$.

allows us to express $\pi(\mathbf{x})$ as a linear function of the independent variables $\mathbf{x}$, $\pi(\mathbf{x}) = G(\mathbf{x}^{'}\beta)$, which yields the final form of our logistic regression model

$$y = G(\mathbf{x}^{'}\beta) + e. \tag{3.54}$$

Note that the inverse of $G(\mathbf{z})$, called *logit transformation*, is denoted by

$$\text{logit } \pi(\mathbf{x}) = \ln \frac{\pi}{1-\pi} = \mathbf{x}^{'}\beta. \tag{3.55}$$

3.2.2 Maximum Likelihood Estimation of the Logit Model

The general method of estimation of logit models is maximum likelihood [38], in which we basically want to maximize the so-called *likelihood function* to find an estimator agreeing most closely with the observed data. Consider the model (cp. equation (3.54))

$$\mathbf{y} = G(\mathbf{X}\beta) + \mathbf{e} \tag{3.56}$$

where $\mathbf{X} \in \mathcal{R}^{n\times(p+1)}$, $\beta \in \mathcal{R}^{p+1}$ are the design matrix, including the column of the constants, and coefficient vector, respectively, along with $\mathbf{y} \in \{0,1\}^n$ denoting the response variable. Note that $\mathbf{y}$ is mutually independent but not identically Bernoulli distributed, because the probability of success and failure depends on $\mathbf{x}_i$. Finally, as before, e_i equals either $1 - G(\mathbf{x}_i^{'}\beta)$ if $y_i = 1$ or $G(\mathbf{x}_i^{'}\beta)$ if $y_i = 0$, thus $\mathbf{e}$ has mean zero and heteroscedastic variance: $G(\mathbf{X}\beta)^{'}[\mathbf{1} - G(\mathbf{X}\beta)]$. The conditional likelihood function can be written as a product of the n univariate probability densities[21]

$$\begin{aligned} L(\beta|\mathbf{x_i}, y_i) &= \prod_{i=1}^{n} \mathrm{P}(y_i|\mathbf{x_i}, \beta) \\ &= \prod_{i=1}^{n} G(\mathbf{x}_i^{'}\beta)^{y_i}[1 - G(\mathbf{x}_i^{'}\beta)]^{1-y_i}. \end{aligned} \tag{3.57}$$

However, it is mathematically easier to work with the log of equation (3.57), termed as (conditional) *log-likelihood*

$$l(\beta|\mathbf{x_i}, y_i) = \ln[L(\beta|\mathbf{x_i}, y_i)] = \sum_{i=1}^{n} \left\{ y_i \ln[G(\mathbf{x}_i^{'}\beta)] + (1 - y_i) \ln[1 - G(\mathbf{x}_i^{'}\beta)] \right\}. \tag{3.58}$$

To find the optimal β we differentiate equation (3.58) with respect to β and set the resulting expression to zero, yielding the so-called *scoring* equations

$$\mathbf{X}^{'}(\mathbf{y} - G(\mathbf{X}\widehat{\beta})) = 0 \tag{3.59}$$

[21] The constant binomial term has been disregarded.

which constitute not only a necessary but also sufficient condition for $\hat{\beta}$ being the maximum likelihood estimator of β, so it can easily be shown that the log-likelihood function $l(\beta)$ is a concave function. For the remainder of the thesis we will denote the maximized log-likelihood with $l(\mathbf{y}, G(\mathbf{X}\hat{\beta}))$ or just with $l(\hat{\beta})$. It is worth noting, since the first component of each $\mathbf{x}_i$ is one, that the first equation of (3.59) specifies $\sum y_i = \sum G(\mathbf{x_i}\widehat{\beta})$, hence the expected number of ones matches the observed number. Although being very similar to its least squares counterpart, equation (3.59) is non-linear in β and iterative methods are required to solve it. For instance, employing the *Newton-Raphson algorithm*[22] [8], which tries to find the optimum through repeated linear approximations, may lead to the following iterative update scheme [57, 71, p. 708, p. 373]

$$\begin{aligned}\beta^{t+1} &= \beta^t - \left(\frac{\partial^2 l(\beta)}{\partial\beta\partial\beta'}\right)^{-1}\frac{\partial l(\beta)}{\partial\beta} \\ &= \beta^t + (\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}\mathbf{X}'(\mathbf{y} - G(\mathbf{X}\beta^t))\end{aligned} \tag{3.60}$$

with $\mathbf{V} = diag\{\frac{\partial G}{\partial\beta}(\mathbf{x_i}'\beta^t) = G(\mathbf{x}_i'\beta^t)[1 - G(\mathbf{x}_i'\beta^t)]\}$ being a weight matrix[23] and completed by a starting value β^0 and a convergence criterion. Re-expressing the Newton step with $\mathbf{z}_i = \mathbf{x}_i'\beta^t + (y_i - \mathrm{E}[y_i])/\mathrm{Var}(y_i) = \mathbf{x}_i'\beta^t + (y_i - G(\mathbf{x}_i\beta^t))/v_{ii}$ yields

$$\beta^{t+1} = (\mathbf{X}'\mathbf{V}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}\mathbf{z} \tag{3.61}$$

which is referred to as *iteratively reweighted least squares* (IRLS), because each iteration solves a weighted least squares problem $\mathbf{V}\mathbf{X}\beta = \mathbf{V}\mathbf{z}$. The elements of $\mathbf{z}$ are often called the adjusted dependent covariates. Note that the Hessian matrix $\Phi = \mathbf{X}'\mathbf{V}\mathbf{X}$[24] has other uses as well. Its expected value with reverse sign is called the *Fisher information matrix*

$$I = -\mathrm{E}(\Phi). \tag{3.62}$$

Hence, the use of Newton's method to find the solution to the conditional log-likelihood equation is also referred to as *Fisher scoring*.

In the context of logistic regression it is essential to allude the terms of *odds* and *odds ratios* as counterparts to probabilities and probability ratios. Suppose the event A has probability π and so the complementary event A^c has probability $1 - \pi$. Then the odds are defined as the ratio of these two probabilities.

[22] The Newton-Raphson algorithm, which is named after the mathematicians Newton and Raphson, is a well-known technique used in numerical analysis when one wants to find the zero(s) of a function taking real values.

[23] In fact, the diagonal elements of $\mathbf{V}$ equal the variance terms of the error $\mathbf{e}$ and response $\mathbf{y}$, respectively.

[24] Bear in mind that all methods based on the iterative structure discussed above have problems in case of a near singular Hessian matrix, due to the necessary calculation of the inverse.

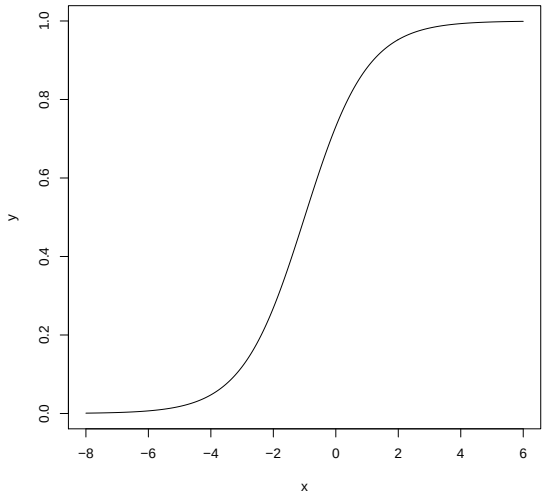


Figure 3.2: Logistic function with one attribute $\mathbf{x}$.

$$\text{odds}(A) := \frac{\text{P}(A)}{\text{P}(A^c)} = \frac{\pi}{1-\pi}. \tag{3.63}$$

While probabilities are confined to the interval $[0, 1]$, odds range over the positive half of the real number axis. By taking logarithms we obtain the log odds, which are usually allowed to take the values $-\infty$ and $+\infty$ in order to establish an one-to-one relationship with the respective probabilities 0 and 1. Note that in case of logistic probabilities the odds are

$$\text{odds}(\mathbf{z}) = \frac{\exp(\mathbf{z})/(1+\exp(\mathbf{z}))}{1-[\exp(\mathbf{z})/(1+\exp(\mathbf{z}))]} = \exp(\mathbf{z}).$$

Thus, for the log odds we get

$$\log \text{odds}(\mathbf{z}) = \log \exp(\mathbf{z}) = \mathbf{z}.$$

Finally, the odds ratio, as the name suggests, is simply the ratio of the odds of two events and may be considered as a measure of the effect size describing the degree of association between two events. For the sake of clarity and a better understanding of the basic idea behind odds ratios we shall consider the following example. Assuming a logit model with only one binary covariate, we may define $X = 1$ for "wearing a ski helmet" versus $X = 0$, if the skier does not wear a helmet. The response variable Y equals one, if a skier sustained severe head injuries from a ski accident and zero, if the head injuries were not severe, thus Y comprises only accidents with head injuries. The underlying data, taken from the *Freizeitunfallstatistik 2007, Kuratorium*

	$Y = 0$	$Y = 1$
$X = 0$	1363	1625
$X = 1$	1475	697

Table 3.1: 2×2 contingency table.

für Schutz und Sicherheit, is summarized in table 3.1. However, keep in mind that the number of the accidents with helmets is an approximate, because unfortunately the exact number was not listed in the statistic. Nevertheless, we may calculate the odds like the following: $\text{odds}(Y|X = 0) = 1625/1363 = 1.19$ and $\text{odds}(Y|X = 1) = 697/1475 = 0.47$, respectively, and the odds ratio is $\text{odds}(Y|X = 0)/\text{odds}(Y|X = 1) = 1.19/0.47 = 2.53$. The result may be interpreted that the risk of a severe head injury not wearing a helmet is 2.53 times as great as for other skiers wearing helmets.

3.2.3 Penalized Logistic Regression

Analogous to ridge and lasso regression, where the objective was to minimize a penalized residual sum of squares, we may incorporate a penalty term to the log-likelihood function for the same purpose of achieving a stable, as well as accurate, regression model from higher-dimensional data. The so-called penalized log-likelihood [22, 28, 54] function may be generally written as follows

$$l_p(\beta_0; \beta; \lambda) = -l(\beta_0; \beta) + \lambda J(\beta) \tag{3.64}$$

where $l(\beta_0; \beta)$ denotes the unrestricted log-likelihood function phrased in (3.58), λ is the regularization parameter controlling the amount of shrinkage and $J(.)$ is a penalty function on the coefficient parameter β. Note that the parameter β_0, the intercept, is not penalized explicitly. There exist various approaches to choose the penalty function, for instance, $J(\beta) = \sum_{j=1}^{p} \beta_j^2$ is directly leading to the ridge approach, whereas $J(\beta) = \sum_{j=1}^{p} |\beta_j|$ to lasso. Some further detailed insights on the choice of the penalty term are discussed in [4], though it should explicitly be pointed out that choosing a "proper" penalty function is not trivial. Ridge penalties do not go hand in hand with the need for variable selection!

In a similar fashion, as in the non-penalized case, the Newton-Raphson algorithm may be applied to equation (3.64) to obtain an estimator of β in an iterative way. However, the result depends heavily on the choice of λ, as outlined in section 3.1.5. For $\lambda \to 0$ we obtain unconstrained logit estimates, as with higher regularization imposed (increased λ) we can reduce the effective size of the model through reducing the effective degrees of freedom (cp. equation (3.45) and [24, p. 66]). Appropriate techniques for choosing λ are, amongst others, cross validation introduced in subsection 3.1.5 or methods based on the *AIC* (Information Criterion of Akaike) or *BIC* (Bayesian

Information Criterion of Schwarz), which shall be further discussed in the following section.

Logistic regression with L1 penalization (3.65) has been introduced and applied by many researchers.

$$\widehat{\beta}^{lasso} = \underset{\beta}{argmin} \{-l(\beta_0; \beta) + \lambda ||\beta||_1\} \tag{3.65}$$

Hastie and Park [54], for instance, developed an algorithm that implements the so-called *predictor-corrector method* of convex optimization to determine the entire path of the coefficient estimates as λ varies, i.e., to find $\widehat{\beta}^{lasso}(\lambda) : 0 < \lambda < \infty$. In the next subsection I give a brief description of the predictor-corrector method for penalized logistic regression as implemented in the contributed R-package `glmpath`.

Predictor-Corrector Method

The predictor-corrector method, related to the Newton-Raphson algorithm, is a technique for solving nonlinear equations that are traced through an one dimensional parameter. Hastie and Park [54] employed this method for tracing the curve $H(\beta, \lambda) = 0$ through λ in $p + 2$ - dimensional space with $\beta \in \mathcal{R}^{p+1}$ (β_0 included) and $\lambda \in \mathcal{R}^+$, where $H(\beta, \lambda)$ is the differential of the log-likelihood function $l_p(\beta; \lambda)$ with respect to β under the assumption that none of the components of β are zero.[25]

$$H(\beta, \lambda) = \frac{\partial l_p(\beta; \lambda)}{\partial \beta} = \mathbf{X}'(\mathbf{y} - G(\mathbf{X}\beta)) + \lambda \operatorname{sgn}\begin{pmatrix} 0 \\ \beta \end{pmatrix}^{26} \tag{3.66}$$

Thus, we want to find an unique solution $\beta(\lambda)$ that minimizes the convex[27] function $l_p(\beta, \lambda)$ for each $\lambda \in \mathcal{R}^+$. For a proof of the existence of such mappings $\lambda \rightarrow \beta(\lambda)$ see, for example, *Implicit Function Theorem* [52, p. 940].

The predictor-corrector method may roughly be summarized in the following steps. In the first step the λ - *step length* is determined. Given some λ_k, the next smallest λ at which the active set changes, denoted by λ_{k+1}, is approximated. The active set is a synonym for the set of coefficients being non-zero. The second step is called *predictor step*, in which the corresponding change in β, due to the decrease in the λ value, is linearly approximated. In the third step, termed as corrector step, an exact solution of $\beta(\lambda_{k+1})$ has to be iteratively obtained. The linear approximate from the predictor step is taken as starting value for the algorithm, so the computational cost for the exact solution reduces to a minimum, because we do not have to make an

[25] Otherwise we would encounter problems with the $||.||_1$ - term, since $f(x) = |x|$ is not differentiable at $x = 0$.

[26] The 0 corresponds to the intercept β_0.

[27] If $f(x)$ is concave then $-f(x)$ is convex; $l(.)$ vs. $l_p(.)$.

initial guess of the starting value. Finally, the current active set is reevaluated, which may lead, if there are possible changes in the set, to a reiteration of the corrector step.

Elastic Net Estimator

In case of strong collinearities of the columns of $\mathbf{X}$, the predictor-corrector algorithm and related iterative schemes might get unstable [54]. More precisely, the Hessian matrix of $l_p(.)$ (see app. A.2) calculated in the predictor step might be ill-conditioned. To overcome this problem it was suggested, originally proposed in [74], to incorporate a quadratic penalty term to equation (3.65). Accordingly, differentiation with respect to β yields

$$\widetilde{H}(\beta, \lambda_1, \lambda_2) = \mathbf{X}'(\mathbf{y} - G(\mathbf{X}\beta)) + \lambda_1 \operatorname{sgn}\begin{pmatrix}0\\ \beta\end{pmatrix} + \lambda_2\begin{pmatrix}0\\ \beta\end{pmatrix} \tag{3.67}$$

where $\lambda_1, \lambda_2 > 0$. This modification goes by the name of *elastic net* and is considered as a compromise between ridge regression and lasso, meaning that elastic net selects variables like the lasso, i.e., more coefficients will be zero, and shrinks the coefficients of correlated covariates like ridge regression [24, p. 73]. In particular when $p >> n$, i.e., the number of covariates is much bigger than the number of observations, Zou and Hastie [74] showed that the elastic net is a promising extension of lasso. The latter's limitations lie in group selection[28] and simply in the nature of convex optimizations problems. Which means that lasso can select at most n ($p > n$) variables before it saturates, also implying that it can not have more than n variables with non-zero coefficient in the regression equation [18].[29] Recently, Friedman et. al. [25] introduced a *coordinate descent* approach for elastic net.[30] A linear combination of the lasso and ridge term was suggested as penalty function. For instance, λ_1 is replaced by $\lambda \cdot \alpha$ and λ_2 by $\lambda \cdot (1 - \alpha)$, i.e., if α was close to one we would obtain lasso characteristics and for $\alpha = 0$ the ordinary ridge penalty. Thus, as α increases from zero to one, the sparsity of the solution increases monotonically to the sparsity of the lasso solution [25, p. 4].

$$\widehat{\beta}^{elastic-net} = \underset{\beta}{argmin}\left\{-l(\beta_0; \beta) + \lambda[\alpha||\beta||_1 + (1 - \alpha)||\beta||_2]\right\} \tag{3.68}$$

In coordinate descent [8], as the name suggests, the objective function is minimized along one coordinate direction at each iteration. Depending on in which order the coordinates are chosen (for example cyclical or greedy [72]),

[28] If there is a group of variables among which the pairwise correlations are high, then the lasso tends to select only one variable from the group and does not care which one is selected (for example dummy variables). See [74, sec. 2.3].

[29] Of course, the number of different variables ever to have entered the model can be greater than n.

[30] An implementation is available in the R - package *glmnet*.

slightly different variants of the algorithm exist. This approach was recently re-discovered to the field of data mining, where its computational speed[31] is quite remarkable. Especially the sparsity of the lasso estimates allows a dramatical speedup, since many coefficients are zero, and iterations can be restricted to the active set. Suppose we have estimates for $\widetilde{\beta}_l$ for $l \neq j$, then a coordinate descent step boils down to a partially optimization of the penalized log-likelihood function equ. (3.64) with respect to $\widetilde{\beta}_j$ ($\widetilde{\beta}_j \neq 0$). One can show that the coordinate-wise update reduces to the following form (with standardized design matrix, fixed $\alpha \in [0,1]$ and fixed $\lambda > 0$) [23]

$$\widetilde{\beta}_j \leftarrow \frac{S\left(\sum_{i=1}^n v_{ii} x_{ij}(z_i - \widetilde{z}_i^{(j)}),\ \lambda\alpha\right)}{\sum_{i=1}^n v_{ii} x_{ij}^2 + \lambda(1-\alpha)} \tag{3.69}$$

where the v_{ii}'s are the entries of the diagonal weight matrix $\mathbf{V}$ defined as in equation (3.60), $\mathbf{z}$ denotes the so-called adjusted response defined as in equation (3.61) and $S(\gamma, \delta) = \operatorname{sgn}(\gamma)(|\gamma| - \delta)^+$ represents the soft-thresholding operator (cp. equ. (3.39)). Furthermore, the term $z_i - \widetilde{z}_i^{(j)} = z_i - \widehat{z}_i + x_{ij}\widetilde{\beta}_j$ may be considered as the *partial residual* for fitting β_j. This update procedure can basically be divided into three steps. As a first step, the unpenalized log-likelihood is approximated using iteratively reweighted least squares. Then soft-thresholding is applied to take care of the lasso contribution to the penalty. And finally, a proportional shrinkage is applied for the ridge penalty. Moreover, enclosing this update procedure in two nested loops, an outer loop dealing with a decreasing sequence of values of λ, and an inner loop in which we cycle through all p variables till some convergence criterion is met, would yield the fundamental structure of the pathwise coordinate descent algorithm, as proposed in [25].

3.2.4 Logistic Regression with PCA and PLS

Lasso, as well as principal components and partial least squares, introduced in section 3.1.2 and 3.1.3, respectively, are appropriate techniques for dimension reduction in linear settings. In the following it is shown that it is also possible to extend PCA (and even PLS) to logistic regressions, which shall be further abbreviated as PCLR (Principal Component Logistic Regression). The basic idea is to first employ PCA on the data, and afterwards to build the logistic regression model with the principal components obtained before [2, 51].

Let us use the same notation as in section 3.1.2. The probabilities of success of the logit model as stated in equation (3.56) may be equivalently

[31] For specific run time tests concerning the cyclical/greedy coordinate descent algorithm compared to competing methods, we refer the technically interested reader to [25] and [72].

expressed in terms of all pc's as

$$\mathrm{P}(y_i = 1|\mathbf{x_i}) = \frac{\exp\{\beta_0 + \sum_{k=1}^{p} z_{ik}\gamma_k\}}{1 + \exp\{\beta_0 + \sum_{k=1}^{p} z_{ik}\gamma_k\}} \tag{3.70}$$

or rewritten in matrix form (cp. equation (3.7)) this becomes

$$\text{logit } G(\mathbf{X}\beta) = \mathbf{X}\beta = \mathbf{Z}\mathbf{A}^{'}\beta = \mathbf{Z}\gamma. \tag{3.71}$$

The original parameters of the logit model can be obtained in terms of those of the model that has as covariates all the pc's included. Since, due to the invariance property of maximum-likelihood, $\widehat{\gamma}$ can be estimated from $\mathbf{y} = G(\mathbf{Z}\gamma) + \mathbf{e}$, and then the coefficient vector $\widehat{\beta}$ can be calculated from $\widehat{\beta} = \mathbf{A}\hat{\gamma}$. In order to possibly improve the model, one could just take a subset of the principal components of the original covariates. Let us decompose $\mathbf{Z}\gamma$ as $\mathbf{Z}\gamma = \mathbf{Z}_s\gamma_s + \mathbf{Z}_t\gamma_t$, with $t = p - s$, $\gamma = \{\gamma_1, \cdots, \gamma_s|\gamma_{s+1}, \cdots, \gamma_p\}$, so that the first s pc's are the most explicative ones. After removing the last term the PCLR model can be restated as the following

$$\mathbf{y} = G(\mathbf{Z}_s\gamma_s) + e^*. \tag{3.72}$$

In a similar manner we obtain an estimate for γ_s using maximum likelihood and an estimate for the original parameters by $\widehat{\beta}_s = \mathbf{A}_s\widehat{\gamma}_s$. The main difference between ordinary principal component regression and PCLR is that the estimator $\widehat{\gamma}_s$ in terms of the first s pc's is not the vector of the first s components of the estimator $\widehat{\gamma}$ in terms of all the pc's[32] [2]. Hence, each time we enter or remove a new principal component the model has to be readjusted, resulting in additional computational expenses.

One criticism of PCLR or principal component regression in general is that the pc's are obtained without taking any dependence between response and predictor variables into account. This issue is tackled by partial least squares, introduced in section 3.1.3, defining latent uncorrelated variables (PLS components) and uses them instead of the original covariates in the regression model. In the linear setting the PLS components are obtained by a sequential maximization of the covariance between linear spans of predictor and response variables. However, in the case of logistic regressions, due to the nonlinear link function, conceptually modification are required. For instance, Bastien et. al. [6] proposed a quite pragmatic approach to deal with the dilemma of non-linearity. They introduced a modification to the classical PLS method in which they merely replaced the series of linear regressions for obtaining the PLS components by a series of logistic regressions. A more sophisticated technique was proposed by Marx [50], termed as *iteratively reweighted partial least squares*, and recently adapted by Ding and Gentleman

[32] $\widehat{\gamma}_s = \{\widehat{\gamma}_1(s), \widehat{\gamma}_2(s), \cdots, \widehat{\gamma}_s(s)\} \neq \{\widehat{\gamma}_1, \widehat{\gamma}_2, \cdots, \widehat{\gamma}_s\}$.

[14] to the task of classification based upon gene expression data.[33] Below I want to address Marx's method, henceforth simply denoted by *general partial least squares* and abbreviated as GPLS, in greater detail, because it is one of the models applied in the simulation part. GPLS is more or less an embedding of IRLS into the framework of PLS. Consider a decomposition of the centered $n \times p$ design matrix $\mathbf{X}$ and the adjusted response vector $\mathbf{z} = \mathbf{X}\beta^* + (\mathbf{y} - \mathrm{E}[\mathbf{y}])/\mathrm{diag}\{\mathrm{Var}(\mathbf{y})\} = \mathbf{X}\beta^* + [\mathbf{y} - G(\mathbf{X}\beta^*)]/\mathbf{V}$ in the following form (cp. equ. (3.18) and (3.19)) with $k < p$:

$$\mathbf{X} = \sum_{i=1}^{k} \mathbf{t}_i \mathbf{p}_i^{'} + \mathbf{E}_k \tag{3.73}$$

$$\mathbf{z} = \sum_{i=1}^{k} q_i \mathbf{t}_i + e_k \tag{3.74}$$

where the $\mathbf{t}_i$'s are latent variables, $\mathbf{p}_i$'s are loadings, $\mathbf{q}_i$'s are scalar coefficients and $\mathbf{E}_k$, e_k are residual terms, respectively. The relevant latent variables are obtained by a sequence of iteratively reweighted least squares regressions (with weight matrix $\mathbf{V}$), maximizing the correlation between $\mathbf{X}$ and adjusted response vector $\mathbf{z}$ (rather than working with the original response $\mathbf{y}$) in a weighted metric. After this the original response variable is fitted on the constructed latent variables in a general least squares sense. For a better understanding a slightly modified (the notation was changed for the sake of consistency with our logit model) version of Marx's algorithm has been placed into the appendix A.4.

[33] An implementation of this method can be found in the R-package *gpls*.

Chapter 4

Regression Diagnostics

In the previous chapter I introduced methods for the purpose of developing good predictive relationships between response variables and covariates. The task of validating such predictive capabilities and issues addressing model selection and model assessment, especially for logistic regressions using biased estimation methods [10, 24, 38, 57], shall be discussed in the following.

Model selection has to be understood as an estimation procedure for determining the performance of different models in order to choose the "best" one. In the lasso case that would simply run to a series of estimations over a grid of regularization parameter values λ, and the best λ according to some error criterion is chosen. The assessment of the resulting model is then performed on new data (out-of-sample testing) by calculating the prediction error. A common approach in data-rich situation is to randomly divide the data set into three subsets: a *training set* for fitting, a *validation set*[1] to estimate prediction error for model selection and a *test set* for assessment of the *generalization error* (prediction error over an independent subsample). Such scheme would bypass the circumstance that in case of using the same data for fitting and testing, the prediction error would be underestimated substantially, and usually a tendency towards overfitting the model can be observed. Coming up with a general "dividing rule" is far from being an easy job, by reasons of, among other things, the dependence on the signal-to-noise ratio in the data, the training sample size and the complexity of the model. For instance, a typical split might be 50% of the data used for training and 25% each for validation and testing. Besides quantitative issues, like the question of how many observations each subset should contain, one has to overcome qualitative problems concerning a proper representation of all classes in each subset. For instance, if we consider a binary response vector $\mathbf{y}$ of a logit model where each observation y_i $i \in I_n$ can be classified into

[1] The validation step will be skipped in the simulation part in the next chapter, and I will just divide the data set into a training set for model selection and a test set for model assessment.

responders $y_i = 1$ and *non-responders* $y_i = 0$, it will be essential to have a certain number of responders and non-responders in each subset, otherwise the model will not learn to classify correctly. Such strategies go by the name of *stratification* and are described in [71, p. 590].

4.1 Training Error and Test Error

Given our standard model $y = f(X) + \epsilon$ with $\mathrm{E}(\epsilon) = 0$ and $\mathrm{Var}(\epsilon) = \sigma_\epsilon$ and a training set τ, the *training error* (loss over the training set) may be defined as, following the notation of Hastie et. al. [24, p. 220f]

$$\overline{\mathrm{err}} = \frac{1}{n} \sum_{i=1}^{n} L(y_i, \hat{f}(\mathbf{x_i})) \tag{4.1}$$

where $L(.\ ,\ .)$ is some appropriate loss-function. Moreover, the *test error*, also referred to as *generalization error*, may be written as the prediction error over the test sample

$$\mathrm{Err}_\tau = \mathrm{E}[L(\mathbf{y}, \hat{f}(\mathbf{X}))|\tau]. \tag{4.2}$$

Note that the test error refers to the error for the specific training set, since τ is fixed. A related quantity is the expected test error $\mathrm{Err} = \mathrm{E}[\mathrm{Err}_\tau]$, averaging over everything that is random, including the randomness in the training set. In case of a squared error-loss, the expected test error can be expressed in terms of a bias and variance decomposition

$$\begin{aligned} \mathrm{Err} &= \mathrm{E}[\mathbf{y} - \hat{f}(\mathbf{X})]^2 \\ &= \sigma_\epsilon + [\mathrm{E}\hat{f}(\mathbf{X}) - f(\mathbf{X})]^2 + \mathrm{E}[\hat{f}(\mathbf{X}) - \mathrm{E}\hat{f}(\mathbf{X})]^2 \\ &= \sigma_\epsilon + \mathrm{Bias}^2(\hat{f}(\mathbf{X})) + \mathrm{Var}(\hat{f}(\mathbf{X})). \end{aligned} \tag{4.3}$$

The first term is the variance of the error ϵ and can not be avoided. The second term is the squared bias, the amount by which the average of our estimate differ from the true mean, and the last term represents the variance of our estimate. Typically, an increase in the complexity of a model results in a lower bias but higher variance, in the literature well known as the bias-variance trade-off. In other terms, it is commonly said, that a model with large bias but low variance underfits the data while a model with low bias but large variance tends to overfit the model. However, for the sake of completeness, it should be mentioned that the behavior of the trade-off may depend on the chosen loss-function $L(.\ ,\ .)$ [24]. In particular for classification tasks appropriate loss-functions include the so called *0–1 loss function* (output is either one if correctly classified and zero if not)

$$L(\mathbf{y}, \hat{G}(\mathbf{X})) = \mathrm{I}_{\mathbf{y} \neq \hat{G}(\mathbf{X})} \tag{4.4}$$

and one based on the maximum log-likelihood

$$L(\mathbf{y}, \hat{G}(\mathbf{X})) = -2\ l(\mathbf{y}; \hat{G}(\mathbf{X})) \tag{4.5}$$

where the term "$-2 \times$ maximum log-likelihood" is also known as the *deviance* or *cross entropy*. In logistic regressions with binary response variables the (binomial) deviance reduces to

$$\text{deviance} = -2 \sum_i y_i \log[\hat{G}(\mathbf{x_i})] + (1 - y_i) \log[1 - \hat{G}(\mathbf{x_i})]. \tag{4.6}$$

Various methods have been proposed for the purpose of estimating the prediction error. Few of these techniques are introduced in the next part, divided into empirical and theoretical methods.

4.2 Theoretical Methods for Model Selection

Let us define with *op* the optimism as the difference between the training error $\overline{\text{err}}$ over the training data set $\tau = (\mathbf{y}^{train}, \mathbf{X}^{train})$ and the *in-sample* error Err_{in} over $(\mathbf{y}^{new}, \mathbf{X}^{train})$ (see Hastie et. al [24, p. 229])

$$op \equiv \text{Err}_{in} - \overline{\text{err}}. \tag{4.7}$$

The in-sample error rate is defined as $\text{Err}_{in} = \frac{1}{n} \sum \text{E}_{\mathbf{y}^{new}}[L(y_i^{new}, \hat{f}(x_i^{train}))|\tau]$, meaning that new response values y_i are observed at each of the training points. Furthermore, the average optimism ω is the expectation of the optimism over the training set outcome values with fixed predictors $\omega = \text{E}_{\mathbf{y}}(op)$, and can generally be written as [24]

$$\omega = \frac{2}{n} \sum_{i=1}^{n} \text{Cov}(\hat{y}_i, y_i). \tag{4.8}$$

The last equation simply indicates that the harder we fit the data, the greater the covariance between $\hat{y}_i$ and y_i will be, thus increasing the expected optimism ω. Upon taking the expectation of equation (4.7) we may rewrite it to

$$\text{E}_{\mathbf{y}}[\text{Err}_{in}] = \text{E}_{\mathbf{y}}[\overline{\text{err}}] + \frac{2}{n} \sum_{i=1}^{n} \text{Cov}(\hat{y}_i, y_i) \tag{4.9}$$

which in the linear case[2] for additive models $\mathbf{y} = f(\mathbf{X}) + \epsilon$ can be reduced to [24]

$$\text{E}_{\mathbf{y}}[\text{Err}_{in}] = \text{E}_{\mathbf{y}}[\overline{\text{err}}] + \frac{2d}{n} \sigma_\epsilon^2 \tag{4.10}$$

where d should capture the model complexity. Expression (4.10) is considered as the underlying idea of the information criteria introduced in the next

[2] Versions of (4.9) hold approximately in the non-linear case too.

subsection, which is first estimating the optimism and then adding it to the training error to get an approximate for the in-sample error.[3]

4.2.1 Akaike Information Criterion

The Akaike information criterion abbreviated as AIC was developed by Akaike in 1971 and proposed in [3] for the purpose of statistical identification as a measure of the goodness of fit of estimated models. Akaike suggested to take the log-likelihood estimate as a criterion of the "fit" of a model, by reasons of being a quantity which is most sensitive to deviations of the model parameters from the true values, and add a correction term (similar to the expected optimism in equations (4.9) and (4.10)).

$$\text{AIC} = -\frac{2}{n} l(\mathbf{y}; G(\mathbf{X}\hat{\beta})) + 2 \cdot \frac{d}{n} \tag{4.11}$$

If the maximum likelihood is identical for two competing models it will be the best choice, when focusing on the principle of parsimony, to take the less complex one, i.e., the model with fewer parameters. Since d in equation (4.11) acts as a measure of the model complexity or in Akaike's words, d is *the number of independent adjusted parameters*, the model with the lowest AIC score should be preferred. Furthermore, AIC is by definition an asymptotically unbiased estimator of the mean expected log-likelihood, but is in general not asymptotically consistent in terms of selection criterion, i.e., given a family of models, including the true model, the probability that AIC will select the correct model is strictly smaller than one as the sample size $n \to \infty$ [29]. Typically, AIC tends to choose too complex models as the sample size increases, given room for further modifications, as for instance the Bayesian information criterion (BIC).

4.2.2 Bayesian Information Criterion

BIC was developed by Schwarz 1978 [59] and is applicable like AIC in settings where the fitting is done by maximization of a log-likelihood function.

$$\text{BIC} = -2\ l(\mathbf{y}; G(\mathbf{X}\hat{\beta})) + \log(n) \cdot d \tag{4.12}$$

In spite of the similarity to AIC (4.11) (the factor 2 is replaced by $\log(n)$) BIC is motivated differently. More precisely, it it motivated in a Bayesian framework, originating from approximating the evidence ratios of models known as the *Bayes factors* [24, p. 234]. Another difference is that BIC enjoys the consistency property in terms of selecting the true model [73], but often chooses models that are too simple in the finite sample case, due to its heavy penalty on complexity.

[3] That the in-sample error is not exactly brilliant is evident, since future values are not likely to coincide with their training set values, but, especially for model comparison, the in-sample error is a convenient alternative.

4.3 Empirical Methods for Model Selection

Cross validation [9, 24] and Bootstrapping [17, 44] are resampling methods for estimating the expected prediction (generalization) error. Although being numerically intensive, and thus making resampling only applicable in an environment with access to fast hardware, its advantage lies in the conceptually simplicity, i.e., resampling requires fewer assumptions and has a greater generalizability than traditional parametric approaches.

4.3.1 Cross Validation

In k-fold cross validation the data set, further denoted by D, is randomly split into k mutually exclusive subsets $D_1, D_2 \cdots, D_k$ of roughly equal size. Then the model is fitted k times. Each time $t = 1, \cdots, k$ it is fitted on D/D_t and for testing the prediction error is calculated when predicting D_t (cf. equation (3.42)). The type of partitioning (the number of folds) allows a specific classification of cross validation. For instance, $k = n$ goes by the name of *leave-one-out* cross validation (LOOCV), and here, as the name suggests, in each fit one observation is left out from the training data set. However, LOOCV induces not only high computational costs, since n fits are necessary, but also high variance in the cross validation estimate, because the D_i's $i = 1, \cdots, k$ are so similar to one another [24, p. 242]. It is interesting to note that the LOOCV technique is asymptotically equivalent to AIC [63], thus the latter can be used as a "fast and (computationally) cheap" substitute for LOOCV, especially for huge data sets. Decreasing the number of folds (smaller k) may considerably decrease the variance in the estimates (D_i's are less similar), but leads to an increase in biasness (smaller data set D/D_t for fitting). Kohavi [44] recommended to take 5-fold or 10-fold cross validation as a good compromise, and suggested for further bias reduction to use a stratification approach or repeated runs.

4.3.2 Bootstrapping

The bootstrap family was introduced by Efron [17], and analogous to cross validation it seeks to estimate the expected generalization error. Apart from that, the intrinsic difference between these two techniques is that the bootstrap re-samples the available data at random *with* replacement, whereas cross validation does it *without* replacement. Suppose we have a training set τ and want to fit a model to this data. The basic idea of bootstrapping is to randomly draw data sets with replacement from the training data, each sample the same size as τ, and to repeat this process B (e.g. $B = 50$) times, producing B bootstrap data sets S_b $b = 1, \cdots, B$. Then the model is refitted to each of the bootstrap data sets, providing a holistic picture of the behavior over the fits. Common methods for estimating the expected prediction error

include the leave-one-out bootstrap estimate (cp. LOOCV) and the 0.632 estimate, which is an extended and improved, in terms of bias reduction, variant of the former. Let the leave-one-out bootstrap estimate be denoted by $\hat{\text{Err}}^{(1)}$ and defined like the following [24, p. 251]

$$\hat{\text{Err}}^{(1)} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i)) \tag{4.13}$$

where C^{-i} is the set of indices ($|C^{-i}|$ the number) of the bootstrap samples S_b $b = 1, \cdots, B$ that do not contain observation i. Bearing in mind that the probability of any observation not being chosen after n samples is (with $p = 1/n$):

$$\text{P}(\text{obs. } i \notin S_b) = \binom{n}{0} p^0 (1-p)^{n-0} = (1-1/n)^n \approx \exp^{-1} \approx 0.368. \tag{4.14}$$

Hence, the average number of distinct observations in each bootstrap sample is about $0.632 \cdot n$, from which one may infer that leave-one out bootstrapping has similar characteristics (low variance/high bias) as 2-fold cross validation [24, p. 252]. To alleviate the high bias of the last estimate another bootstrap method was proposed, which is referred to as ".632 estimator" ($0.632 = 1 - 0.368$ relates to (4.14)).

$$\hat{\text{Err}}^{(.632)} = 0.368 \cdot \overline{\text{err}} + 0.632 \cdot \hat{\text{Err}}^{(1)} \tag{4.15}$$

The derivation of (4.15) is relatively complicated and is not a necessity for understanding the basic idea behind it, which may be considered to be the attempt to reduce the bias of the leave-one-out bootstrap estimate $\hat{\text{Err}}^{(1)}$ by pulling it towards the training error rate of the training set.

4.4 Measures of Goodness-of-Fit

4.4.1 Receiver Operating Characteristics (ROC)

The ROC [38] graph, originating from signal detection theory[4], is a conceptually simple technique for visualizing, organizing and selecting classifiers based on their performance. In general, a classification model (classifier) is a mapping from instances to predicted continuous (e.g. an estimate of an instances class membership probability) or discrete classes. Let us consider a two-class prediction problem (binary classification), in which the outcomes $\mathbf{y}$ either are "positive" ($y_i = 1$) or "negative" ($y_i = 0$). For instance, if the

[4]One of the first uses was during World War II for the analysis of radar signals. Following the attack on Pearl Harbor in 1941, the United States army began new research to increase the prediction of correctly detected Japanese aircraft from their radar signals.

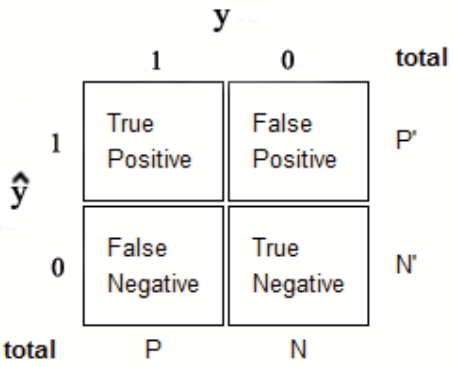


Figure 4.1: 2×2 confusion matrix.

outcome from a prediction is $\hat{y}_i = 1$ and the actual value is also $y_i = 1$, then it is called a true positive (tp). In figure 4.1 the four possible outcomes from a binary classifier are illustrated, from which several common metrics may be derived.

$$true\ positive\ rate = \frac{\text{Positives correctly classified}}{\text{Total positives}} \tag{4.16}$$

$$false\ positive\ rate = \frac{\text{Negatives incorrectly classified}}{\text{Total negatives}} \tag{4.17}$$

$$specificity = \frac{\text{True negatives}}{\text{False positives} + \text{True negatives}} \tag{4.18}$$

The ROC graph plots the probability of detecting true signal (true positive rate on the y-axis) versus a false signal (false positive rate on the x-axis) for an entire range of possible cut points[5], and may also be considered as a tool for visualizing the trade-off between benefits (*tp rate*) and costs (*fp rate*). Points in the "north west" of the ROC space reflect higher *tp rate* and lower *fp rate* and are therefore regarded as desirable.[6] For instance, if the objective is to maximize the classification by choosing an appropriate cut point, one would select a cutoff level that maximizes the correctly and minimizes the incorrectly classified outcomes. The ROC graph, however, is as a quantitative measure of predictiveness of competing models only useful to a limited extent. A common alternative way is to calculate the area under the ROC curve (AUC), which ranges from 0 to 1 and provides a scalar measurement of the classifier's ability to discriminate. Generally spoken, $0.7 \leq AUC < 0.8$ is

[5] As the cut points vary, the elements of $\hat{\mathbf{y}}$ are different classified. E.g. cut point: 0.5: If $\hat{y} \geq 0.5 \rightarrow \hat{y} = 1$ else $\hat{y} = 0$.

[6] Simple random class guessing would produce a diagonal line in the ROC space.

considered as an acceptable and $0.8 \leq AUC$ as an excellent discrimination. For a deeper insight into this topic, such as multi-class ROC graphs or ROC curve averaging (cp. cross validation), we refer the interested reader to [20].

In concluding this chapter, I particularly want to point out that seeking the "true" model is searching for an impossibility. Rather it is important to seek for a model, which is easily understandable[7], parsimonious, appropriate for the situation and which gives a plausible approximation to reality.

[7] That is usually not the case with biased estimation methods, what should not bar us from using them for estimation purposes in the next chapter.

Chapter 5

Real Data Analysis

Thankfully, the data set for the real data tests was provided by a mail order company, consisting of purchase history information of 68210 customers. The main objective is to select customers who should receive the winter catalog 2009. All testing and modeling were performed in R 2.9.1 (*http://www.r-project.org/*) and as a computing platform a Quad Core i920 Intel Processor with 6GB system memory was available. The following R-packages were in use throughout the analysis: `ROCR` [61], `GLMNET` [26], `PLS` [68], `GPLS` [13] and `gregmisc`.

5.1 Data set Description

The characteristics of each individual customer is summarized in 388 continuous and categorical candidate covariates, ranging from buying behavior (products, product quantities, ...) to product specific variables (product prize, ...) and to customer's personal data (age, gender, ...), causing a quite messy design matrix $\mathbf{X}$. Categorical variables, in general, can be grouped into quantitative discrete and qualitative ordinal/nominal variables (cp. figure 5.1). Powers and Xie [56] describe the distinction between quantitative and qualitative as that the former measurements *closely index the substantive meanings of a variable with numeric values, whereas numerical values for qualitative measurements are substantively less meaningful*. The quantitative discrete category includes variables which may only assume integer values. The covariates *bas0_ YYaXX*[1], for instance, the number of orders of a specific product type (XX stands for the product type) from a single customer during a predetermined time period (YY stands for the season) falls into this group. The second group comprises variables such as *kaufkraftkl*, which indicates the class of purchasing power and goes from one to nine or *ortsgroesse*, classifying the customers according to the size of the city where

[1]Discussing all covariates in detail would go beyond the scope of this thesis.

Quantitative { Continuous ⇒ Non-Categorical; Discrete ⇒ Categorical }

Qualitative { Ordinal ⇒ Categorical; Nominal ⇒ Categorical }

Figure 5.1: Typology of the four types of measurement.

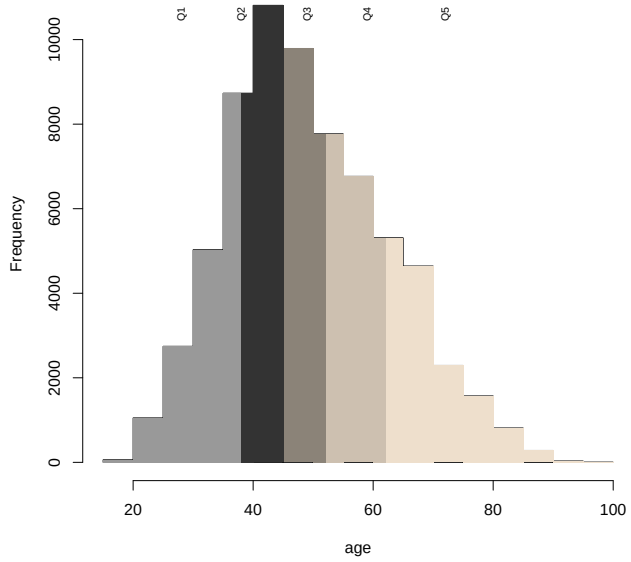


Figure 5.2: Histogram of variable *age* (*alter1*) with colored quintiles.

they are living. Qualitative nominal variables, on the other hand, possess no inherent ordering. Examples are *gender* or *marital status*.

Transformation of covariates is commonly applied in regression analysis, when the variance or mean of the original variables differ to any significant degree. I decided to standardize (center and scale) all continuous candidate variables, due to the different units and keep the categorical candidate covariates unchanged. The variable *age*, although being considered as conceptually continuous, is often categorized or discretized, i.e., it is divided into subintervals of, for example, 5 years and each subinterval is represented by a categorical covariate. The length of these intervals and where to place the

cut points, although appearing to be straightforward at first glance, should be chosen wisely. I also want to follow this discretization approach for the variable age (*alter1*) and use the quintiles[2] as cut points. To be able to state the full model we need observed outcomes as well, which are further combined in the binary response vector $\mathbf{y}$. If for an individual customer i the outcome takes the value one ($y_i = 1$), then this person has ordered anything during the recorded purchase history, and if customers have not ordered anything, then the corresponding outcome takes the value zero. It should also be mentioned that the data set for each test setup has been split into a training set and a test set. The size of the former is about 2/3 of the whole data set and the remaining observations are used for the final model assessment. To ensure diversity across the population of the sets a stratification approach, stratifying on response, has been applied.

5.2 Method Description

5.2.1 Principal Component Logistic Regression (PCLR)

On the one hand, one would like to capture most of the variance in $\mathbf{X}$ and don't "loose" too much information by replacing the covariates with a smaller number of principal components, but on the other hand, for the sake of parsimony, it is essential to retain as few principal components as possible. Some rules for this balancing act are summarized in [42] and [66]. Further, I want to focus on cross validation, already discussed in the context of empirical methods for model selection sec. 4.3, and also applicable for selecting a significant subset of principal components. For instance, the cross validatory scheme presented by Eastment and Krzanowski [46] tries to quantify the idea of "acceptable accuracy" in terms of prediction of individual elements of $\mathbf{X}$. The scheme hinges on the fact that each element x_{ij} of $\mathbf{X}$ is predicted from an equation like the SVD, but based on a submatrix of $\mathbf{X}$ that does not include x_{ij}. The sum of squared differences between predicted and observed x_{ij} acts as measure of predictive accuracy and may be written as

$$\text{PRESS}(m) = \sum_{i=1}^{n} \sum_{j=1}^{p} ({}_m\hat{x}_{ij} - x_{ij})^2 \tag{5.1}$$

where ${}_m\hat{x}_{ij}$ is predicted from a m-components model. A valuable alternative from a computational point of view and especially for huge data matrices would be to divide $\mathbf{X}$ into k subsets (cp. k-fold CV) instead of calculating each $\hat{x}_{ij}$ separately. Hence, predicting the kth set from the projection matrix constructed from the remaining $k-1$ sets [35, 40]. In general, PRESS(m) is expected to fall as m increases, since the predicted values will become more

[2] Quintiles divide the distribution into fifths. For example, the first quintile divides a distribution such that 20% of the observations lie below it.

accurate as more pc's are included into the model. So PRESS by itself is not a satisfying selection criterion. In order to remedy to this problem Wold [70] suggested an additional statistic (R-statistic)

$$R(m) = \frac{\text{PRESS}(m)}{\sum_{i=1}^{n}\sum_{j=1}^{p}\big((m-1)\hat{x}_{ij} - x_{ij}\big)^2}. \tag{5.2}$$

This compares the PRESS after fitting m components with the sum of squared differences between observed and estimated data points based on all the data using $(m-1)$ components. Important components were felt to be those for which $R < 1$. It should be mentioned, however, that even the R statistic is not the last conclusion of wisdom and the unity cutoff level should be considered to be taken with cautiousness and flexibility. As an alternative criterion for an appropriate selection of the pc's an approach developed by Mertens et al. [53] was implemented (for the pseudo code see appendix A.5). They use a version of PRESS equation (5.1) which also takes the response variable $\mathbf{y}$ into account by calculating the *mean-squared error of probabilities* (MSEP). It is defined as the following

$$\text{PRESS-MSEP}(m) = \sum_{i=1}^{n}(y_i -_m \hat{y}_i)^2 \tag{5.3}$$

where ${}_m\hat{y}_i$ denotes the estimate of y_i obtained from a PCLR based on a subset m and using $\mathbf{X}$ with the ith observation deleted. The third criterion, here presented due to its wide use and conceptually simplicity, is called *Kaiser's rule*. This criterion accepts all eigenvalues above the average eigenvalue and rejects those below the average eigenvalue. That is why it also referred to as *Average Eigenvalue Rule*. The average eigenvalue is $1/p \cdot tr(\Sigma)$, which is, for correlation-based PCA, equal to 1. Additionally, it should be mentioned that for PCLR to work properly the entire (categorical covariates too) design matrix $\mathbf{X}$ has to be centered, due to technical reasons.

5.2.2 General Partial Least Squares (GPLS)

From a modeling point of view general partial least squares seems to be most challenging. Not only because the underlying theory is complicated and difficult to understand, but rather because the complexity of the algorithm is a huge computational burden for any hardware and that there is not much literature about GPLS being applied to real data. Especially, it is unusual to deal with such huge and mixed design matrix like in our case. Nevertheless, by setting the convergence criteria of the algorithm for the coefficients relatively moderate and restricting the set of GPLS components to a minimum, one was able to calculate cross validation estimates in a feasible computing time. For the interested reader are the R-source code (App. C) and the pseudo code (app. A.4) placed into the appendix. Additional

convergence problems[3] were eluded by enabling the Firth bias reduction option [21]. Firth developed a procedure to remove the first-order term of the asymptotic bias of maximum likelihood estimates based on a modification of the score function (3.59). Furthermore, binomial deviance- and 0-1 loss function are employed for computing the GPLS cross validation estimates. I will, henceforth, denote the cross validation error with 0-1 loss function simply by misclassification rate and the other one by deviance.

5.2.3 Elastic Net

Elastic net testing is focused on three different values for the "mixing" parameter of the penalties, namely $\alpha = 0.05$, $\alpha = 0.5$ and $\alpha = 1$. From an optimization point of view, of course, one could work with a range of α-values and take the value which performs best in terms of an appropriate error criterion. Let us reminisce that $\alpha = 1$ corresponds to lasso, $\alpha = 0.5$ to a linear combination of the lasso and ridge penalty and $\alpha = 0$ to ridge regression. For the latter, however, instead of exactly 0 a small value was chosen. First, because of algorithmic stability reasons and second, that ridge normally shrinks the variables but do not set them zero. So all covariates, although with small magnitude, would enter the regression, and this is precisely what is not intended. The second tuning parameter regarding elastic net issues is the regularization parameter λ. This parameter is due to the enormous time necessary for computing, especially in a cross validation scheme, restricted to a set of 20 different values for each α. These values are chosen uniformly on the log scale covering the entire range only limited by the size of the active set[4] (the covariates with non-zero coefficients). Which is set to 280 again due to computing feasibility, i.e., values for λ where the number of non-zero covariates exceed 280 are not taken into account. Finally, for measuring the prediction error, serving as basis for determining the optimal model, 10-fold cross validation estimates with binomial deviance- and 0-1 loss function are calculated. Note that if one also wanted to use α as a tuning parameter, cross validation on a 2D surface would be necessary (see sec. 5.3.1).

[3] For instance, due to complete separation [14], i.e., there exists a vector that correctly classifies all observations (perfect prediction), thus the maximum likelihood estimator does not exist. Especially with high-dimensional data this problem is common.

[4] The size of the active set can be considered as an estimate for the degree of freedoms of the model [75].

5.3 Simulations

5.3.1 Test Scenario 1

In the first scenario a random sample of size 10000 without replacement is chosen from the population. The rate of responders in the sample is 49%, i.e., 4900 customers out of the 10000 were potential buyers. The training set for parameter tuning consists of 6666 customers, which is about 2/3 of the sample size, and the remaining individuals are kept for the test set. Due to stratification is the rate of responders in the training sample about the same as in the entire sample: 48.48%.

Parameter Tuning

First of all we shall have a closer look at the eigenvalues of the sample training covariance matrix. The value of the largest eigenvalue is $l_1^2 = 3.14 \cdot 10^3$, whereas the smallest eigenvalue with $l_{392}^2 = 1.07 \cdot 10^{-29}$ may be considered to be numerically zero. Moreover, the last 89 eigenvalues are smaller than $1 \cdot 10^{-27}$, which may be interpreted as a significant sign of collinearities between the covariates. The condition number of the sample covariance matrix, which is defined as the positive square root of the ratio of the largest to the smallest eigenvalues, adds up to the enormous value of $1.17 \cdot 10^{16}$ and is, without doubt, another indication of an ill-conditioned matrix. Bear

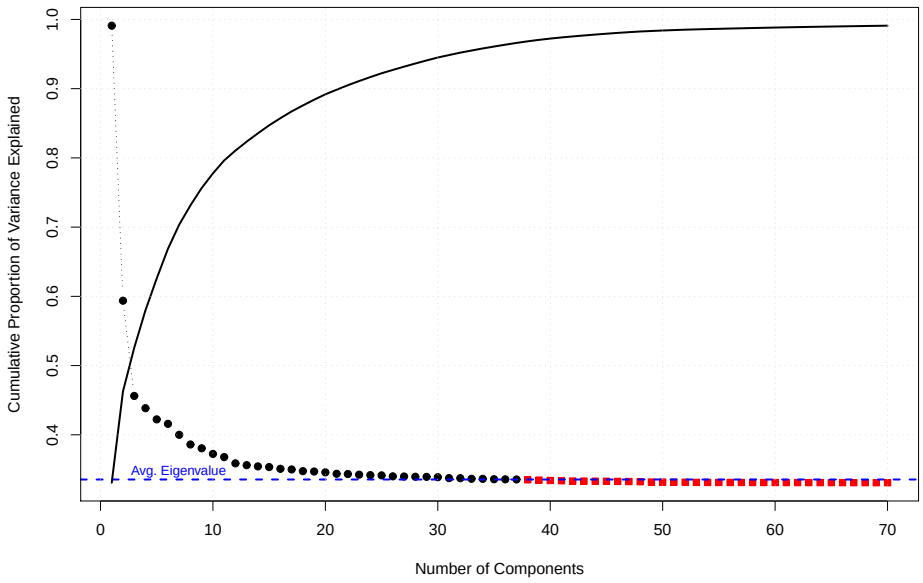


Figure 5.3: Eigenvalues of the sample training covariance matrix.

in mind that the eigenvalues of the covariance matrix equal the variances of linear combinations[5] of the covariates. The average eigenvalue rule, introduced in subsection 5.2.1, recommends that only eigenvalues above the average eigenvalue are worth considering.[6] The value of the mean eigenvalue acting as threshold in the training sample is 24.4. Therefore, the criterion would retain the first 37 principal components accounting for 96.6% of the total variation in the data. As a second "stopping" criterion the prediction sum of squares statistic PRESS(m) is taken into account, which is evaluated by 10-fold cross validation. The corresponding curve, demonstrated in figure 5.4, is monotonic decreasing for all $m = 1 \dots 80$. Therefore, this approach seems unfeasible for retaining an optimal[7] number of components. As exposed by [46] and also experienced by [35], departures from monotonicity tend to be slight in practice.

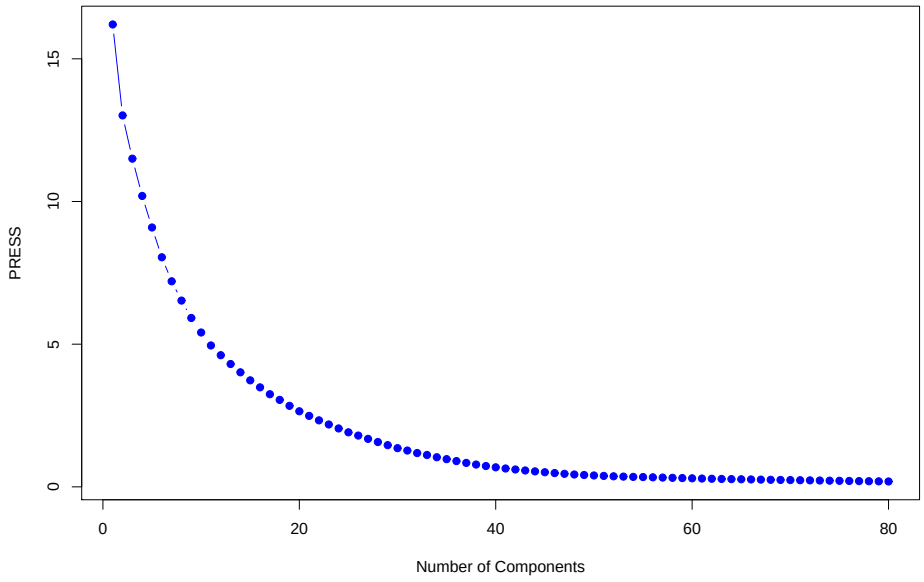


Figure 5.4: PRESS evaluated by 10 fold cross validation and averaged over 20 consecutive runs.

Nevertheless, in certain circumstances it could be argued that removing principal components from the regression equation where the slope of the PRESS curve is small or near constant (i.e., the gain of information through

[5] As per definition, principal components are linear combinations of the predictor variables.

[6] Illustrated in figure 5.3: Black filled circles are pc's above avg. eigenvalue; red filled squares are pc's below avg. eigenvalue.

[7] Minimum of the press curve.

adding additional pc's into the regression equation is marginal) is feasible. For example, regarding figure 5.4, one may retain only the first 50 principal components. However, I would strongly recommend to take alternative criteria into consideration. Such as the R-statistic presented by S. Wold [70], which is more or less built by the fraction of the PRESS statistic and the residual sum of squares of $\mathbf{X}^{train}$ (see equ. (5.2)). The R-statistic calculated with included pc's ranging from 30 to 80 is illustrated in figure 5.5, where pc's below the cutoff level (black filled circles) are felt to be the important ones and pc's above the cutoff level (red filled squares) may be omitted. The first 52 pc's are, according to this method, worthwhile to be kept, which explain about 98.75% of the total information in the training matrix. It should be mentioned that the unity cutoff level was suggested by S. Wold and may situationally be modified.

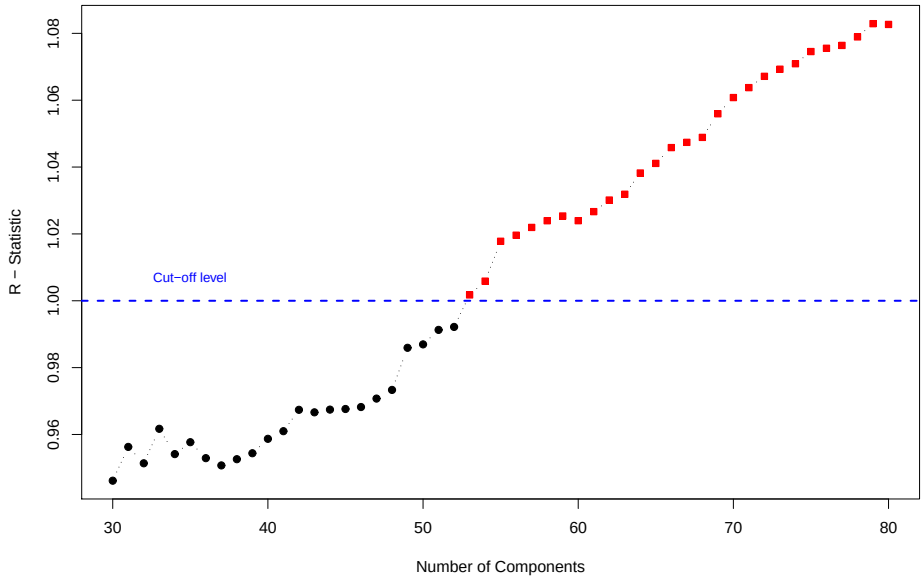


Figure 5.5: R - Statistic.

The last criterion of interest is the so-called PRESS-MSEP. Contrary to the two previous methods, this approach works with the response variable by computing the mean squared error (L2 error) between the predicted probabilities and the observed outcome. Considering figure 5.6, the minimum averaged L2 cross validation error occurs at 56 components, i.e., in case of using the first 56 pc's for predicting purposes, the resulting model has lower L2 error than any other PCLR model in a 10-fold CV estimation scheme. The length of the two bars at each point equals the standard deviation of

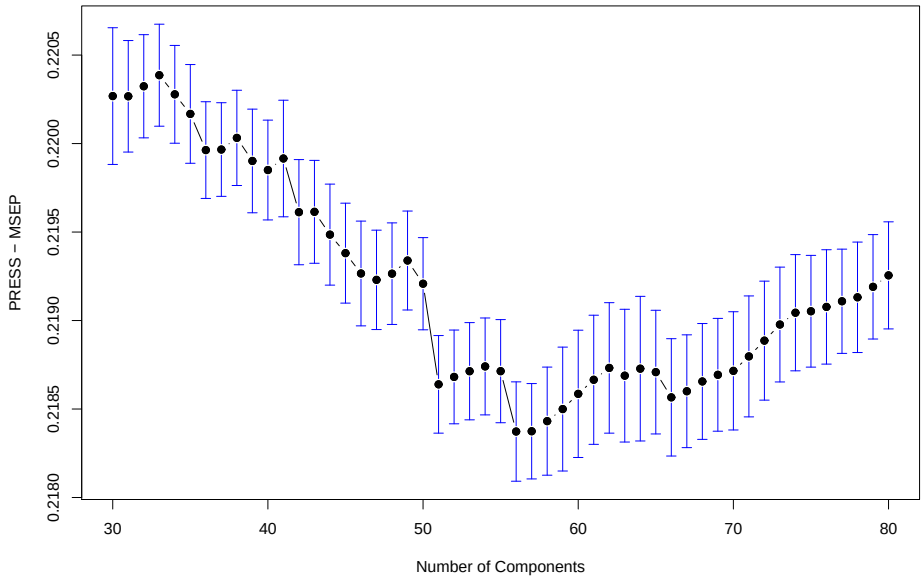


Figure 5.6: PRESS-MSEP evaluated by 10 fold cross validation and averaged over 20 consecutive runs.

the CV estimates and may be considered as the significance bound. As it can be seen, there is no relevant significant difference between a model with 51 pc's and 56 pc's included, because the mean error for the former lies at the upper end of the significance bound of the latter. In such issues one should always choose the most parsimonious model. Hence, for the PRESS-MSEP criterion the optimal number of components is 51, accounting for 98.45% of the total variance. Let us summarize the three presented methods: The avg. eigenvalue rule would suggest to retain the first 37 pc's, on the other hand, the R-statistic would choose the first 52 pc's and the PRESS-MSEP discussed in the end would take the first 51 pc's into the final regression model. Surprisingly, the optimal numbers of components proposed by the last two methods are contiguous, which is, at least in my opinion, a good argument to enter the final testing for PCLR with about 50 pc's. In fact I take excactly 50 pc's.

Let as further examine the linear combination of the first principal component, the most explanative one. Due to the huge set of variables only the covariates with coefficients greater than 0.1 are printed below. The full linear combination can be found in the appendix B.2.1. Needless to say, since not all covariates have been scaled to have variance one (categorical variables remained unchanged), the following selected variables may possibly

not represent the most important ones. It should be just an example of how appropriate analysis could be applied to the pc's, and it shows that pc's are not appearing from nowhere.

```
    rqs00     rqs03     rqs05     rqs06     rqs08   badob_8
0.2220719 0.3274624 0.3973186 0.3725015 0.3303159 0.1021969
  badob_9   batex_7   batex_8   batex_9
0.1262883 0.1023475 0.1443726 0.1806760
```

The variables *rgs00/03/05/06* and *rgs08* seem to play a significant role, where *rgs* refers to the rate of returns of products and the last two figures stand for the specific time period.[8] Furthermore, *badob_8* and *badob_9* capture information about the number of products the customer has ordered from the assortment of ladies' fashion, and the covariates *batex_7, batex_8* and *batex_9* include not only ladies' fashion but also men's fashion and kids' fashion.[9] Interestingly, all these variables are also included in the set of optimal candidate covariates determined by the three elastic net models.

The general partial least squares procedure implemented in the R-package `gpls`, attesting to its complexity, may be referred to as a black box with input $\mathbf{X}$, $\mathbf{y}$ and output $\hat{\mathbf{y}}$, the fitted probabilities. The internal structure, even though a detailed pseudo code is presented in appendix A.4, is somewhat difficult to understand and as a technique for interpretation purposes to a great extent inappropriate. Hence, it is not a surprise that partial least squares is considered to be an algorithm than anything else. As discussed previously, (general) partial least squares seeks for components that explain as much of the relationship between $\mathbf{X}$ and $\mathbf{y}$ as possible. Similar to PCLR, these components are then used for the further modeling process. The procedure implemented in the R-package `gpls`, however, does not allow to explicitly extract information about the general partial least squares components, thus no detailed analysis can be applied on the components. The procedure returns just the entire coefficient vector $\hat{\beta}_s^{gpls}$ (s = number of GPLS comp.), which has already been converted back to the realm of the original covariates. This issue rises the problem of determining the degrees of freedom of the model correctly. Since, regardless of how many GPLS components are used to calculate $\hat{\beta}^{gpls}$, the prediction equation always contains the full set of covariates. And as Marx [50] argues, it may be improper to simply use the number of covariates as a measure for the degrees of freedom.

[8] "00":=fall/winter season 2008; "03":=spring/summer season 08; "05":=s/s 08 - f/w 08; "06":=f/w 07 - f/w 08; "08":=f/w 06 - f/w 08.

[9] "_07":=f/w 07 - f/w 08; "_08":=s/s 07 - f/w 08; "_09":=f/w 06 - f/w 08.

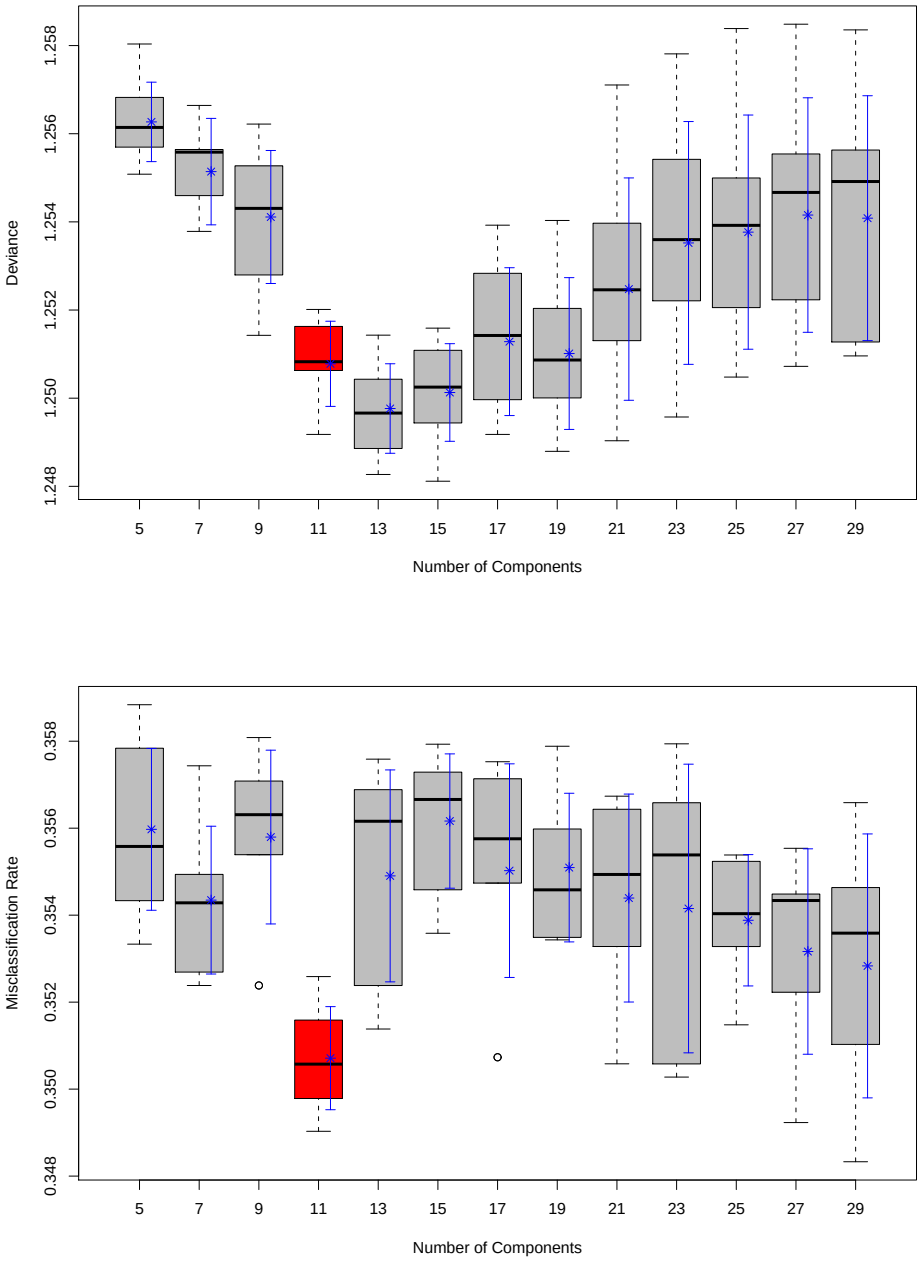


Figure 5.7: General partial least squares training tests: (top) Boxplot of 10-fold CV estimates with binomial deviance loss function over 10 consecutive runs; (bottom) Boxplot of 10-fold CV estimates with 0–1 loss function over 10 consecutive runs.

The training results for GPLS are presented in figure 5.7, where boxplots[10] of 10-fold CV estimates with binomial deviance loss function (equ. (4.6)) and with 0–1 loss function (equ. (4.4)) are plotted. It seems, when inspecting figure 5.7, that there is no statistically significant difference in cross validation error (deviance) between models having 11 gpls components included and models with more components. Due to the parsimony issue, I choose as optimal tuning value 11 components, which is confirmed by the fact that the selected model greatly outperforms the rest in terms of misclassification rate. Preferred are obviously candidate models with low deviance and low misclassification rate. The latter one should be believed only partly, though, because the cutoff level was fixed to 0.5, i.e., outcome probabilities greater than 0.5 are classified with ones and otherwise with zeros. A better performance measure are the ROC curves employed to final testing, due to the consideration of an entire interval of cutoff levels. Throughout the training step I will consider the deviance as the main selection criterion, while taking the classification only to some extent into account.

The graph on the top in figure 5.8 shows the coefficient paths of elastic net regression with $\lambda = 0.05$. The red dashed line marks the optimal parameter value determined by 10-fold cross validation. A low λ value puts more weight to the ridge penalty as to the lasso penalty, which means that in general, as already several times discussed, the covariates are mainly shrunken but not set exactly zero. Hence, the solution has more non-zero coefficients than lasso, but with smaller magnitudes. The graph in the middle and on the bottom should illustrate the other case, where ridge penalty and lasso penalty are equally weighted, and the lasso term is on its own, respectively. Friedman et. al. [25] argue that the coefficient profile of the latter shows typically more "wild" characteristics and a scattered, less concentrated distribution. This being one reason for adding shrinkage behavior through an additional L2 penalty to ensure potentially more stable coefficient paths. A comparison of the graphs in figure 5.8 should confirm this finding in practice, at least in the low λ range it is observable. For the sake of clarity (due to the huge number of covariates) I did not label each coefficient path with the corresponding name of the covariate, although, for more detailed analysis, it would indeed be interesting and informative. For instance, the covariate with the biggest absolute coefficient is bas0_4a02 (the number of products the customer has ordered between fall/winter season 06 and fall/winter season 08) over almost the whole range of λ values, and this in all three elastic net coefficient paths.

[10] Recall that the box stretches from the first quartile to the third quartile and the median is shown as a line across the box.

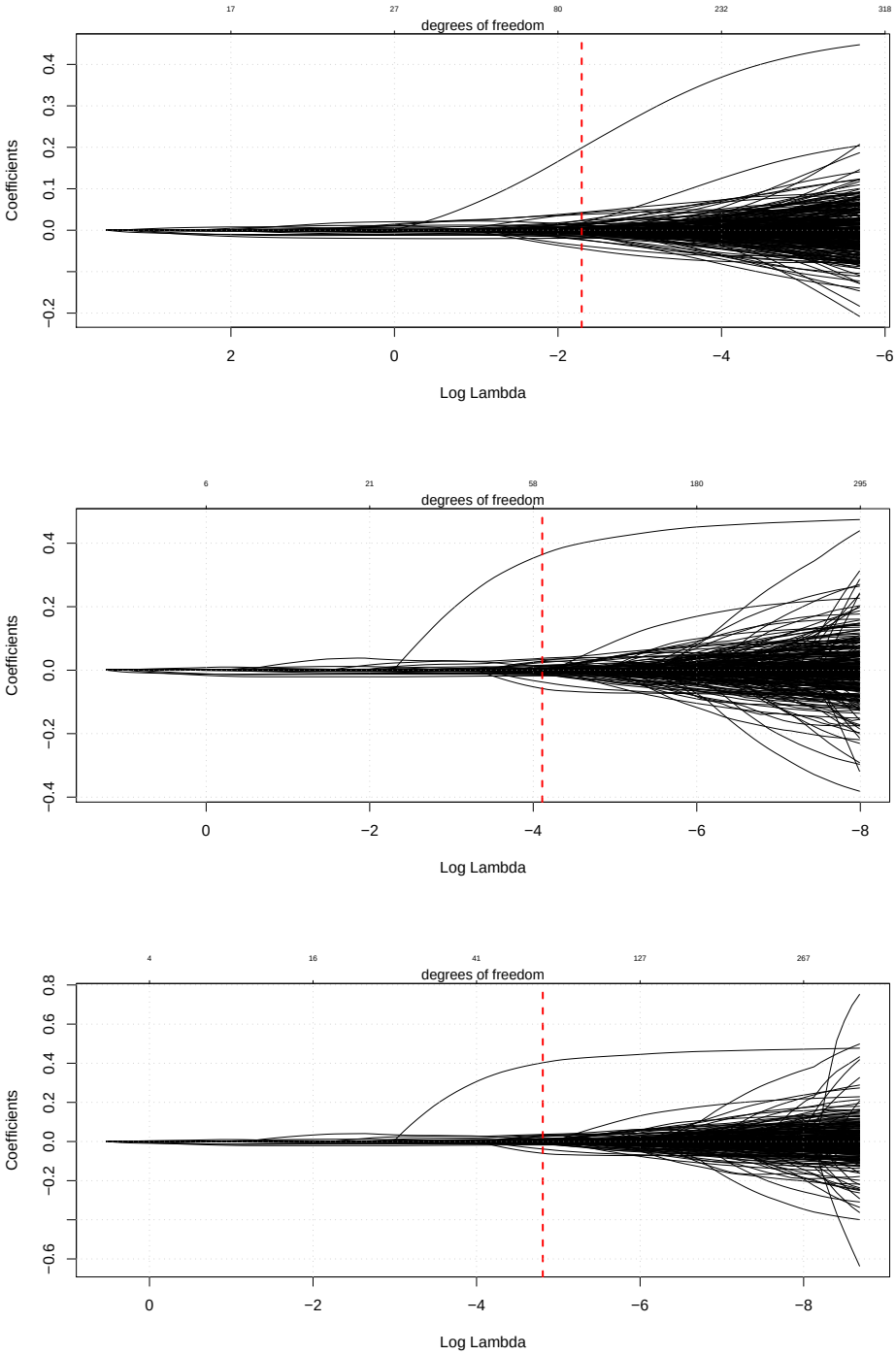


Figure 5.8: Coefficient profile plot of the coefficient paths (training set): (top) Elastic net $\alpha = 0.05$; (middle) Elastic net $\alpha = 0.5$; (bottom) Lasso (elastic net $\alpha = 1$).

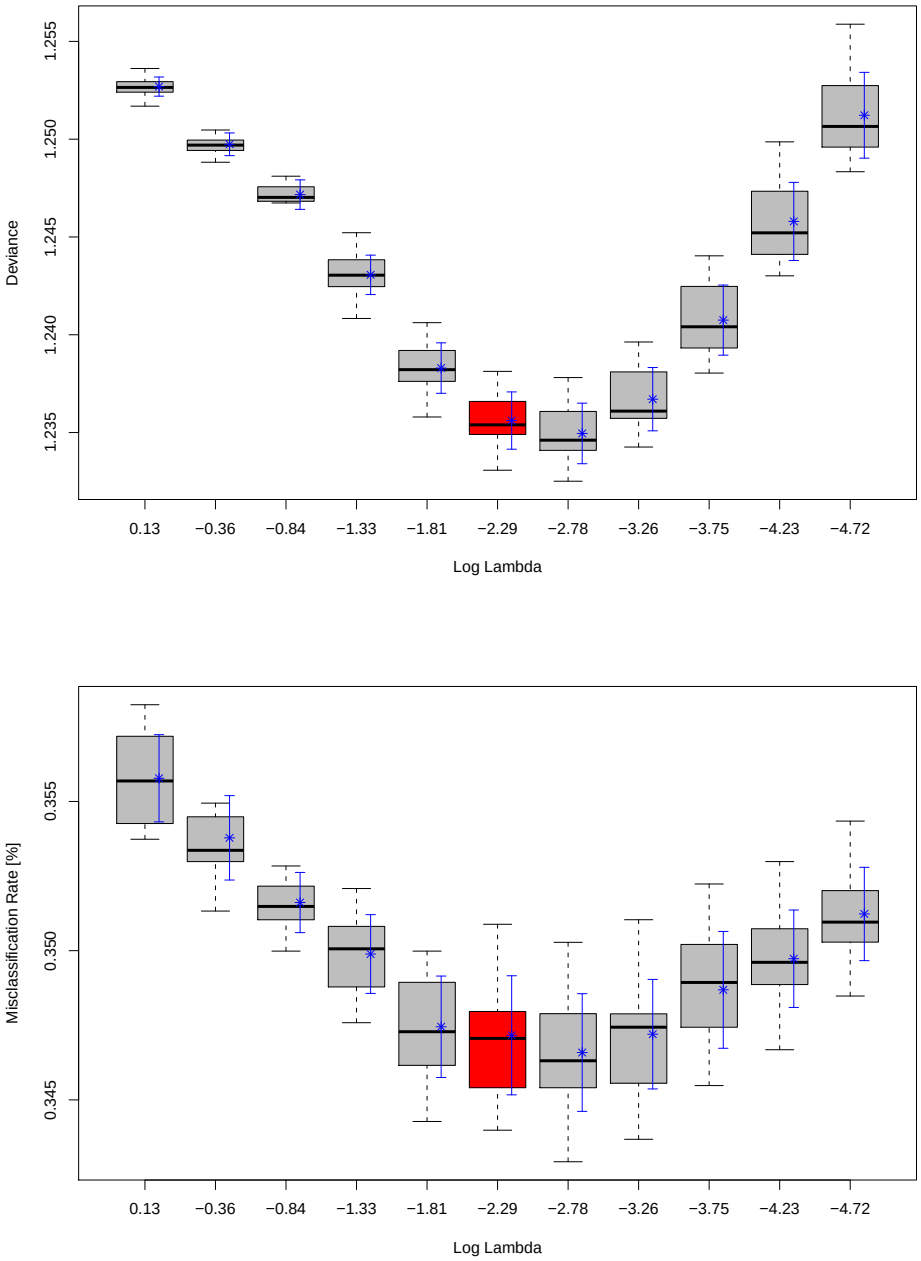


Figure 5.9: Elastic net $\alpha = 0.05$ training tests: (top) Boxplot of 10-fold CV estimates with binomial deviance loss function over 20 consecutive runs; (bottom) Boxplot of 10-fold CV estimates with 0–1 loss function over 20 consecutive runs.

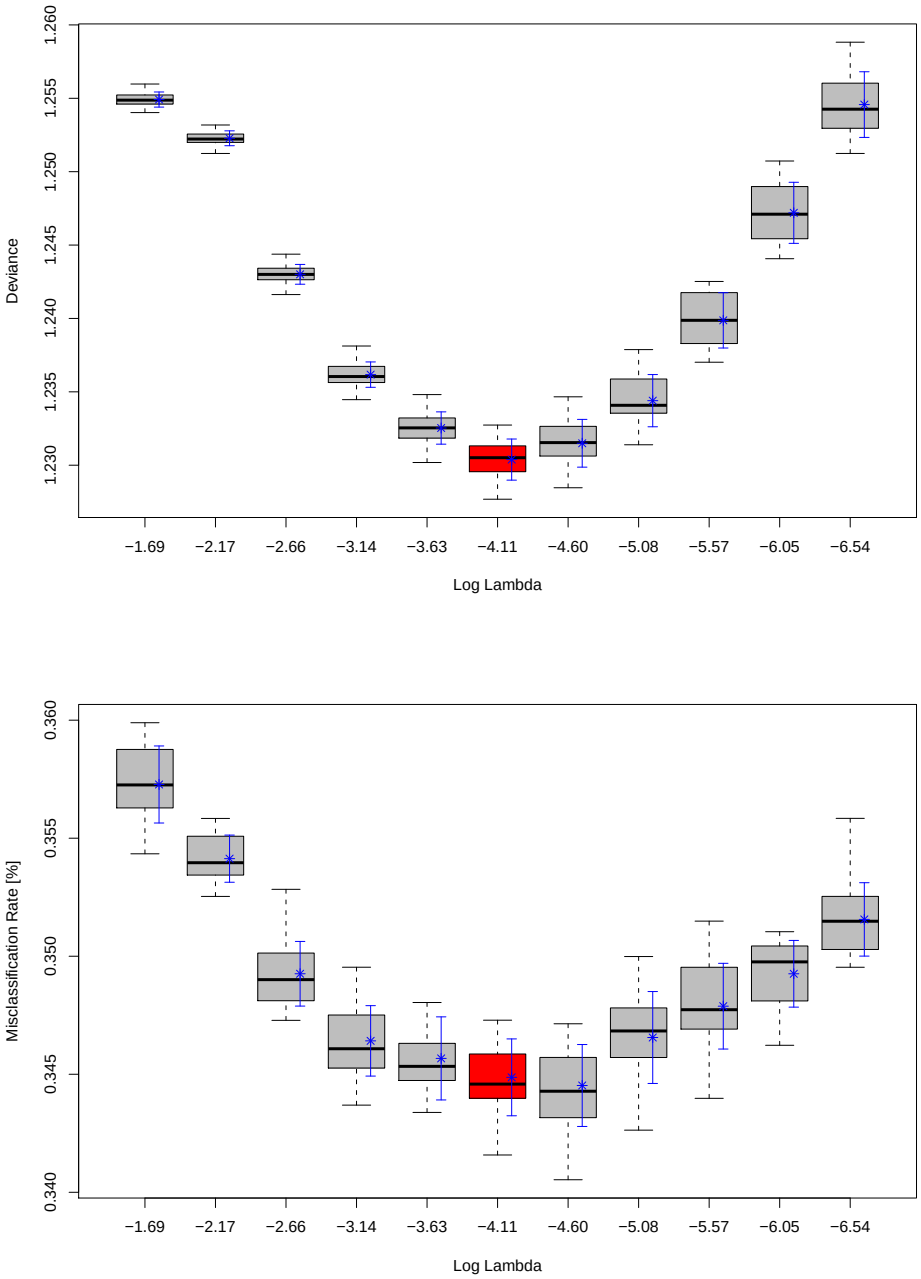


Figure 5.10: Elastic net $\alpha = 0.5$ training tests: (top) Boxplot of 10-fold CV estimates with binomial deviance loss function over 20 consecutive runs; (bottom) Boxplot of 10-fold CV estimates with 0–1 loss function over 20 consecutive runs.

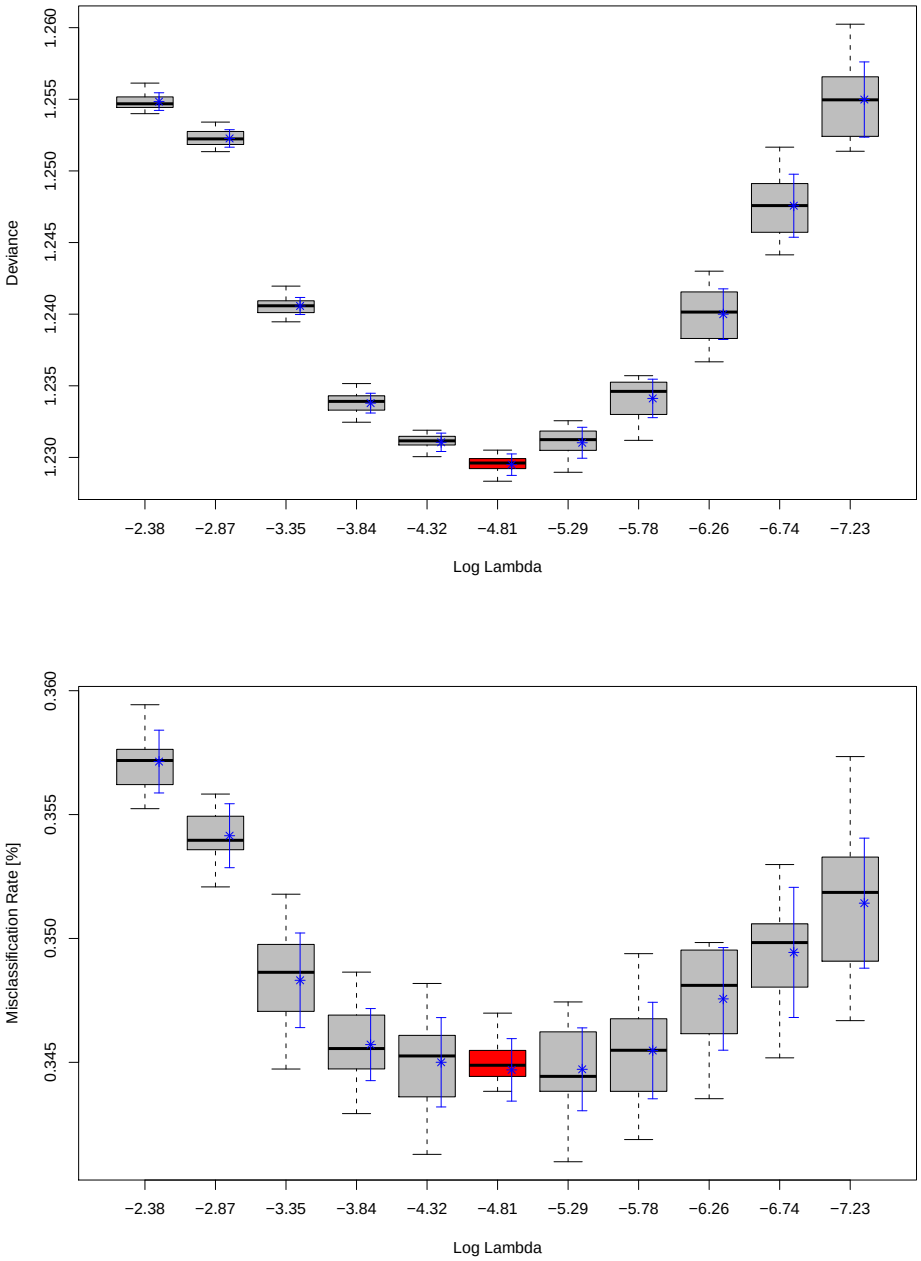


Figure 5.11: Elastic net $\alpha = 1$ training tests: (top) Boxplot of 10-fold CV estimates with binomial deviance loss function over 20 consecutive runs; (bottom) Boxplot of 10-fold CV estimates with 0–1 loss function over 20 consecutive runs.

The optimal tuning parameters may easily be read of the boxplots figures 5.9, 5.10 and 5.11. Apparently, in case of $\alpha = 0.05$, $log(\lambda) = -2.78$ has the lowest mean deviance. Adding the standard deviation of the cross validation estimates to the mean error shows that there is no significant difference in performance between models with $log(\lambda) = -2.78$ and $log(\lambda) = -2.29$, though. Therefore, due to the ambition too choose the less complex one, I take as optimal tuning parameter for the elastic net model with $\alpha = 0.05$ the latter. The best model in terms of misclassification rate is not necessarily easy to clearly identify (fig. 5.9, graph on the bottom), since there are a couple of candidate models with nearly the same mean misclassification rate. At least $log(\lambda) = -2.29$ does not perform significant worse than the others. Moreover, setting $log(\lambda) = -2.29$ results in a model with degrees of freedom of 99, i.e., 99 of the 392 covariates have non-zero coefficients. Obviously, $log(\lambda) = -4.11$ performs best in the $\alpha = 0.5$ case. This value for λ corresponds to a model with degrees of freedom of 62. Our last setup, elastic net with $\alpha = 1$, equals a regression with lasso penalty and according to fig. 5.11 one may suggest to take for the value of the tuning parameter $log(\lambda) = -4.81$ (df = 61).

As one can observe, the complexity of the optimal model reduces with increasing value of α. More precisely, from 99 covariates with $\alpha = 0.05$ to 61 covariates with $\alpha = 1$. In scenario 2 we will make the finding that the optimal models are even more sparser, i.e., the cross validation procedure suggests higher values for the regularization parameter. Furthermore, it is quite interesting to note that lasso and elastic net $\alpha = 0.5$ select almost the same set of candidate covariates. In the latter only the variable *askzs6* (the number of different assortments the customer has ordered from between fall/winter season 07 and fall/winter season 08) is additionally added to the regression equation. For the entire set of chosen covariates by elastic net $\alpha = 0.5$ see appendix B.2.2.

Parameter Testing

For final testing the models were first fit to the training data using the optimal parameters being determined in the tuning step and then applied to predict the test set. The quantitative performances are summarized in table 5.1, and for qualitative results one may have a closer look at the receiver operating characteristic curves 5.13, 5.14 and 5.15 . Furthermore, I plotted a cost curve dealing with unequal cost of misclassification and give an example of expected cost curves introduced by Holte and Drummond [37].

As already discussed, a model with perfect discrimination has a ROC plot that passes through the upper left corner. Therefore, the closer the ROC curve is to the upper left corner, the better. Considering the ROC figures, all our models produce curves above the dashed line, representing a model with simple random class guessing having an area under the ROC curve (AUC) of 0.5. The training ROC curves are, of course, slightly "better"

Table 5.1: Test Results for Scenario 1.

	Mis. Rate	Deviance	AUC	AIC
Principal Component LR	37.28%	1.273	0.6851	1.303
General Partial Least Squares	36.50%	1.267	0.6909	-
Elastic net $\alpha = 0.05$	36.02%	1.253	0.7003	1.312
Elastic net $\alpha = 0.5$	35.72%	1.250	0.7032	1.287
Lasso (elastic net $\alpha = 1$)	35.72%	1.249	0.7036	1.286

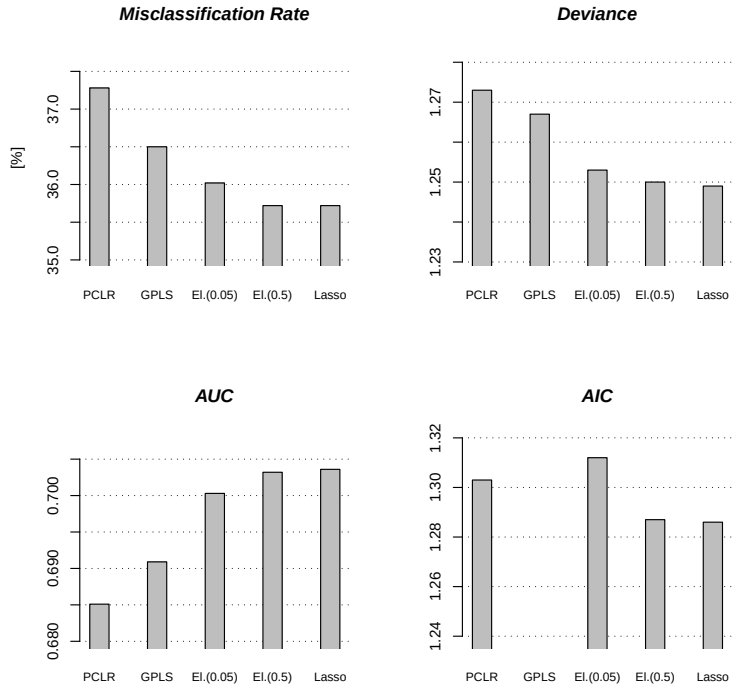


Figure 5.12: Test Results for Scenario 1.

than the test ROC curves, because the models were fitted to the training data set. However, one could state that precisely this slight difference is an indication for a very good predictive ability. Or, to put it differently, a passable trade-off between fitting and predicting can be observed, which can definitely be regarded as the result of the successful parameter tuning through cross validation. In overfitting scenarios, for instance, one could yield near perfect discrimination in the training step, but the prediction

performance would suffer greatly (not always, but in most cases). At first glance it is nearly impossible to find significant differences in the ROC curves of the proposed models. Alternatively, the AUC value in table 5.1 should be taken into consideration. The AUC values, ranging from 0.6851 (GPLS) to 0.7036 (lasso), are not excessively high, but in my opinion acceptable. Furthermore, lasso and elastic net $\alpha = 0.5$ have with 35.72% the lowest misclassification rate. PCLR reachs 37.28% based on the 0.5 cutoff. Even if one would vary the cutoff level, the misclassification rate would not improve seriously (37.01% with 0.49 cutoff), which interestingly also applies to the other models. The AIC score should reflect the trade-off between the fitting ability and the model complexity. Especially the latter is not always easy to work out, which shows GPLS, where I was not able to calculate the AIC score. Because, as also argued by Marx [50] who introduced GPLS, that the model complexity (degrees of freedom) can not be determined with certainty in this context. Lasso has the lowest AIC value (lower is considered to be better), closely followed by elastic net $\alpha = 0.5$ and elastic net $\alpha = 0.05$ trails behind the rest. This corresponds exactly with the complexity of the models, ordered from the lowest to the highest. Summing up, we successfully employed the procedure of cross validation to prevent overfitting in the training step, and we have seen the superiority of L1 penalty over L2 penalty (lasso vs. elastic net α=0.05).

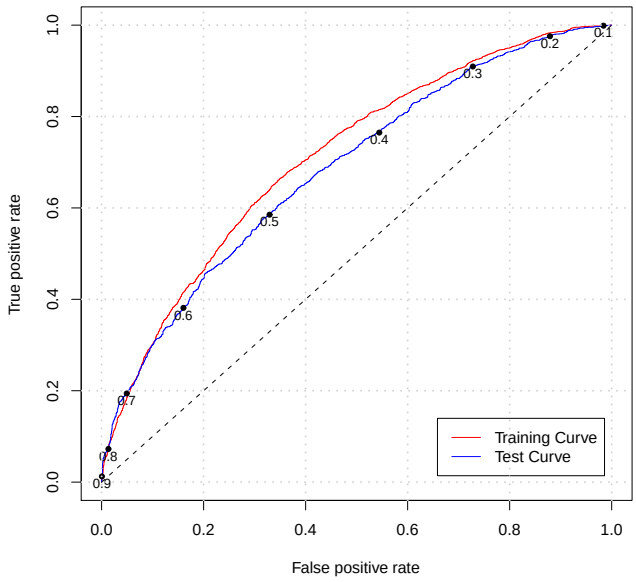


Figure 5.13: ROC training and test curves plus cutoff levels: Principal Component Logistic Regression (PCLR).

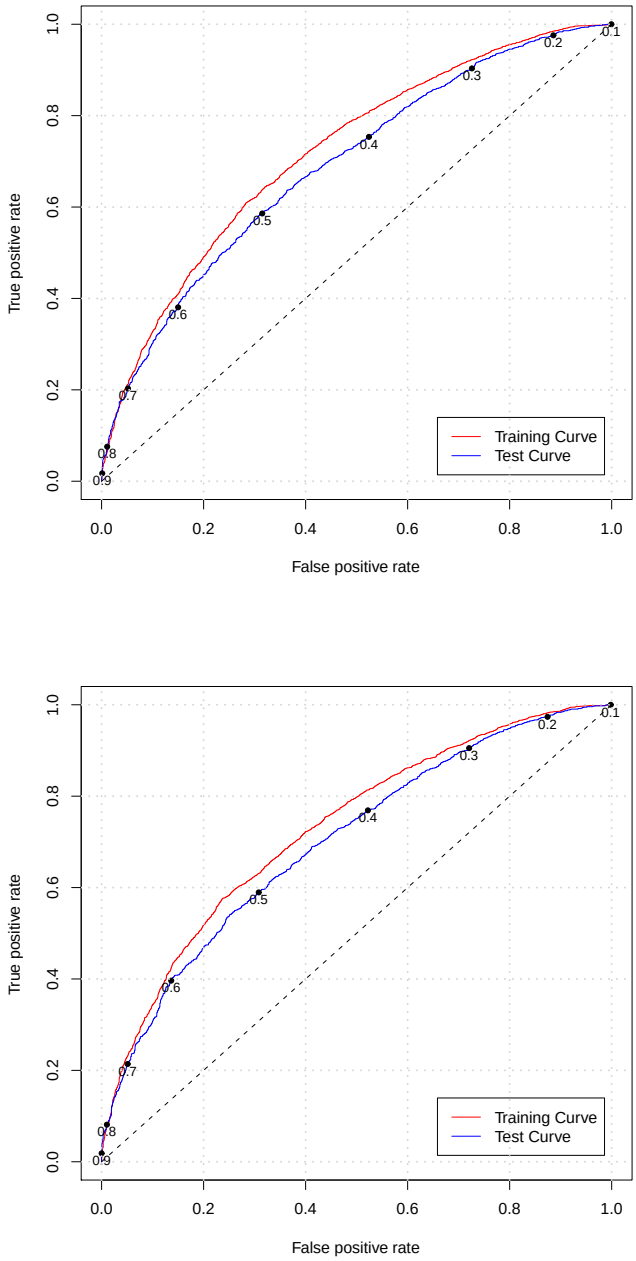


Figure 5.14: ROC training and test curves plus cutoff levels: (top) General Partial Least Squares (GPLS); (bottom) Elastic net $\alpha = 0.05$.

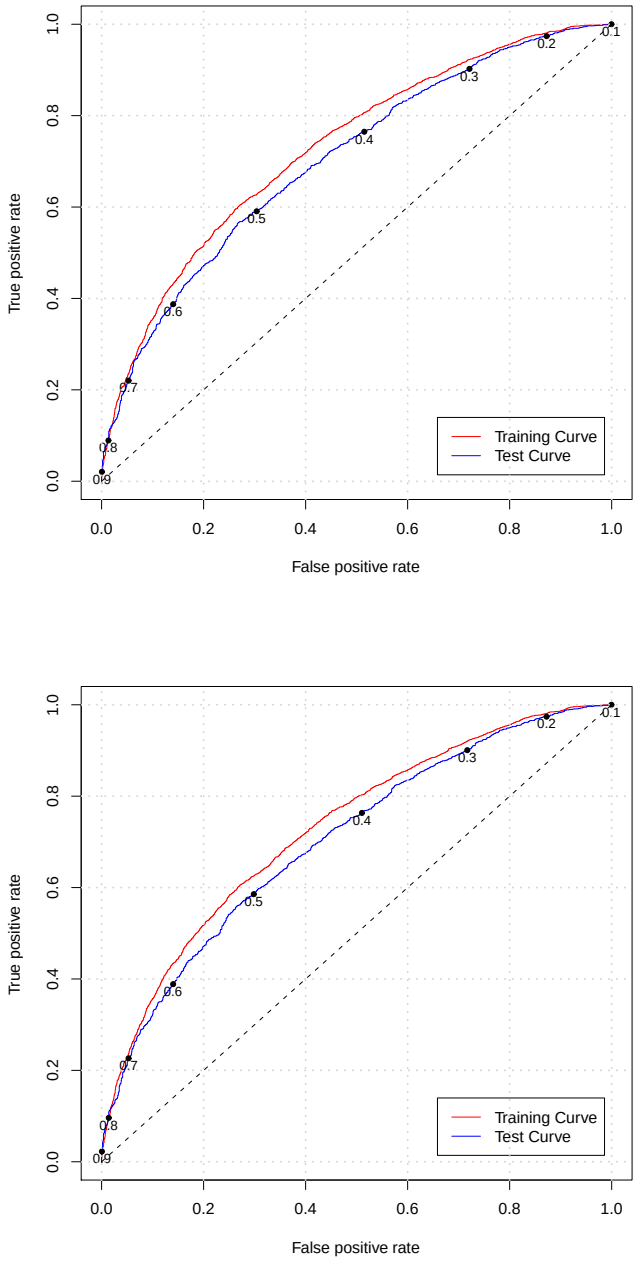


Figure 5.15: ROC training and test curves plus cutoff levels: (top) Elastic net $\alpha = 0.5$; (bottom) Lasso (elastic net $\alpha = 1$).

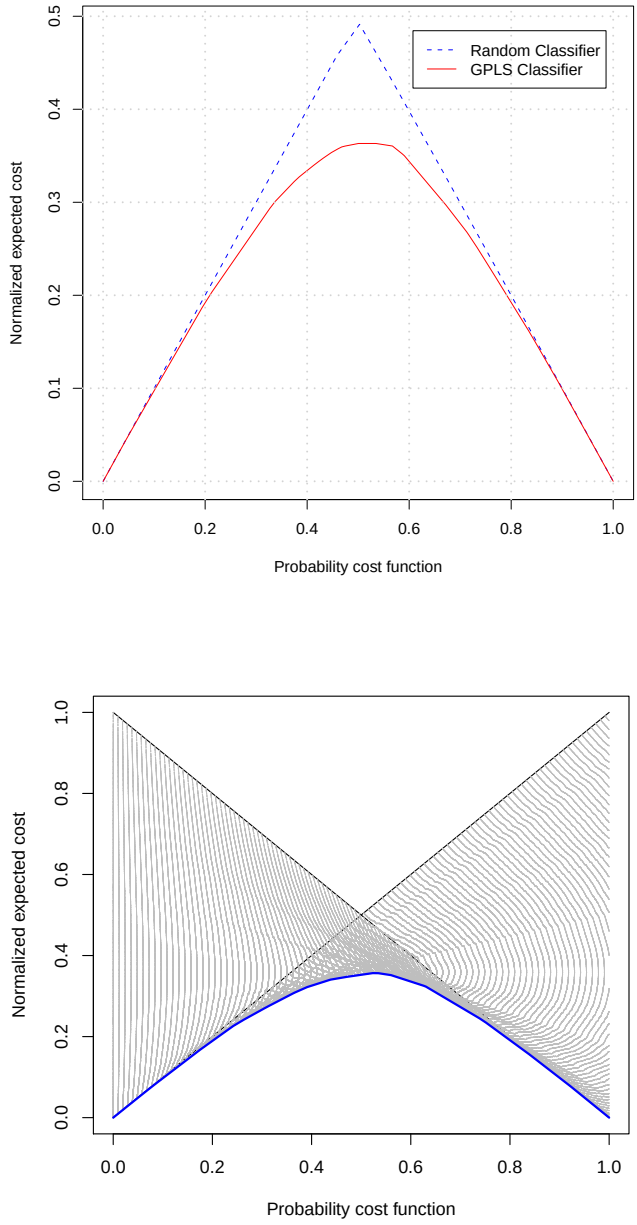


Figure 5.16: (top) Expected cost curve: GPLS classifier vs. random classifier; (bottom) Elastic net $\alpha = 1$: Expected cost curve plus ROC convex hull.

Cost Curves and 2-D Parameter Tuning

Holte and Drummond [37] argue that the ordinary ROC curves are inadequate for the needs of researchers in several aspects. For instance, the ques-

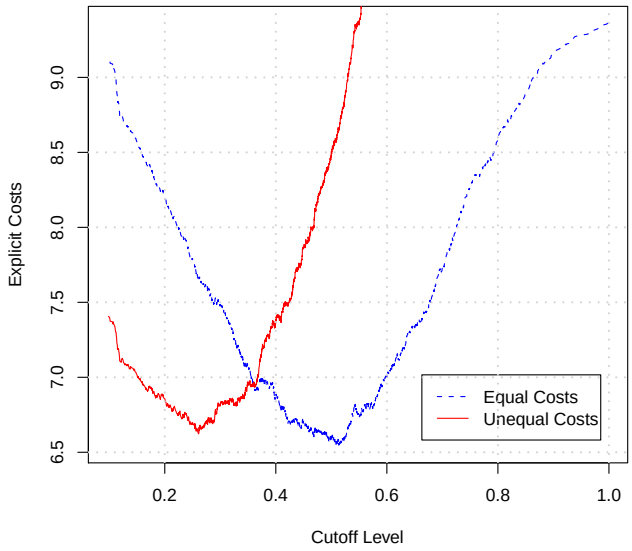


Figure 5.17: Lasso (elastic net $\alpha = 1$: Explicit cost of misclassification per customer.

tion about the cutoff level on which the classifier outperforms the random classifier also called trivial classifier. For this reason a different way of visualizing classifier performance - the expected cost curve - is introduced in [37]. The conception is based on the point/line duality between ROC space and cost space. Remember that each point in the ROC space is determined by a normalized (fp,tp) tuple. The corresponding line in the cost space has $y = fp$ when $x = 0$[11] and $y = 1 - tp$ when $x = 1$, thus the set of points defining the ROC curve become a set of lines in cost space. The expected cost curve is the lower envelope of the given cost lines, due to the objective to minimize the normalized expected misclassification cost for any classifier. These curves are excellent tools for visualizing the performance of competing classifiers, allowing to easily determine the classifier with the least cost at any cutoff level. Note that, as with ROC graphs, the dashed diagonal lines in figure 5.17, joining (0,0) to (1,1) and (1,1) to (0,0), respectively, represent the random classifier performance. Figure 5.16 shows that the GPLS classifier does not perform much better than a random classifier with cutoff levels in $[0, 0.2]$ and in $[0.8, 1]$.

For the sake of simplicity, I have always assumed equal losses when calculating the misclassification rate so far, i.e., I did not distinguish between

[11]The sequence of cutoff levels are plotted on the x-axis, also, in the broader sense, referred to as probability cost [37].

false positives (fp), $\hat{y} = 1$ and $y = 0$, and false negatives (fn), $\hat{y} = 0$ and $y = 1$. However, this is rarely the case in practice. To take account of unequal losses, I calculated the average order price over the last season of the entire population (68210 customers), which is about 45€. Furthermore, one may assume that the cost for one catalog (cost for printing, shipping,...) are not more than 15€. So the misclassification cost for a false negative adds up to 15€ (the mail order company ships a catalog to a customer not willing to buy anything) and for a false positive to 30€ (loss of profit may also be considered as misclassification "cost"). The resulting cost curve for lasso is illustrated in figure 5.17. The explicit costs on the y-axis are calculated as the following: A cutoff level of 0.5 results in 491 false positives and 700 false negatives, hence the cost per customer are $700 \cdot 30 + 491 \cdot 15 = 28365/3334 = 8.51$. According to figure 5.17, minimum cost can be realized with a cutoff level of about 0.25, i.e., due to the fact that false positives are more cost-intensive than false negatives, winter catalogs should already be shipped to customers with a probability of order greater than or equal to 0.25. The misclassification cost of the "equal cost" curve is set to 18.5 € , for visualization reasons only.

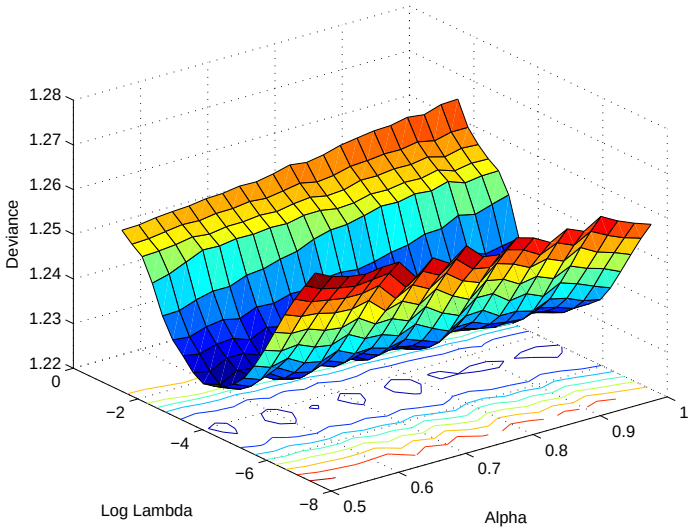


Figure 5.18: Surface and contour plot of 10-fold CV estimates.

Although lasso is in undisputed first place after testing, it seems reasonable to ask the question, if there exist elastic net models with α values smaller than one performing better (inner optimum). For this purpose a surface plot of 10 - fold cross validation estimates with a sequence of 21 values of α between 0.5 and 1 fitted to the training data is illustrated in figure 5.18. As one can see, the deviance is extremely sensitive to λ but relatively insen-

sitive to α. In fact, it is quite a surprise that there is virtually no significant difference in terms of deviance between elastic models with $\alpha \in [0.5, 1]$. One can also observe that when choosing λ too large, values of α closer to 0.5 yield slightly better results. This seems reasonable, since the parameters α and λ are directly related to the model complexity, and the combination of too large regularization term and lasso would produce too sparse solutions.

5.3.2 Test Scenario 2

The second test scenario should examine the 'more predictors than observations' issue and should hopefully provide us with interesting insight into the performance of the different candidate models on such problems. For this purpose, a random sample of size 200 without replacement was chosen from the population. The fraction of responders in the sample is 45,5%, i.e., we have 91 responders and 109 non-responders. The training set size is again about 2/3 of the entire sample size. The rest is used for final model testing. I do not want to address this scenario as detailed as the first scenario. I just want to comment on the optimal tuning parameters and briefly discuss the test results. All necessary graphs can be found in the appendix B.1.

As known, the covariance matrix can have utmost n non-zero eigenvalues in the $n < p$ setup. The value of the largest eigenvalue is $l_1^2 = 3.50 \cdot 10^3$, and the value of the nth-smallest eigenvalue is $l_{133}^2 = 6.90 \cdot 10^{-28}$ ($n = 133$). The next larger eigenvalue is already almost 25 orders of magnitude bigger, $l_{132}^2 = 5.65 \cdot 10^{-3}$. The average eigenvalue rule (fig. B.1) would suggest to keep the first 18 pc's for further modeling. On the other hand, taking the R-statistic and PRESS-MSEP (fig. B.2) into account, three and two components are sufficient to be retained, respectively. Interestingly, in the first scenario the avg. eigenvalue rule proposed to use less pc's for further modeling than the other two criteria and in the second scenario it is the opposite way around. Finally, I decided to pick the first five pc's for testing, explaining about 66,5% of the total variance in the training data set. In the GPLS setup the model of interest seems to have only two components (fig. B.4), performing best in terms of deviance and also yielding an acceptable misclassification rate. The optimal λ's for the elastic net models were obtained the same way as before: Accepting the most parsimonious model including the lowest mean CV error in its significance band. Hence, the values are $log(\lambda) = 2.12$ (df=15), $log(\lambda) = -0.34$ (df=13) and $log(\lambda) = -1.04$ (df=13) for elastic net $\alpha = 0.05$, $\alpha = 0.5$ and $\alpha = 1$, respectively. One should note that the latter two only differ in the coefficients of the covariates (for the set of covariates see B.2.3.).

The test results in table 5.2 reveal that lasso is the best performing model in terms of misclassification rate and deviance. PCLR is found at the end, being not a surprise, since the parameters of the other models were chosen so at to minimize exactly these two criteria. We can also observe the reversed

picture when taking the AIC score into account. Following it, PCLR would be the model of interest. The ROC curves are due to the small sample size (see appendix B.1) relatively difficult to interpret. One might spot that lasso and elastic net $\alpha = 0.5$ have better classifier performance than the other models, especially in the training step.

Table 5.2: Test Results for Scenario 2.

	Mis. Rate	Deviance	AUC	AIC
Principal Component LR	38.81%	1.290	0.6857	1.439
General Partial Least Squares	37.31%	1.266	0.7045	-
Elastic net $\alpha = 0.05$	35.82%	1.225	0.6982	1.698
Elastic net $\alpha = 0.5$	35.82%	1.216	0.7161	1.604
Lasso (elastic net $\alpha = 1$)	34.33%	1.203	0.7196	1.591

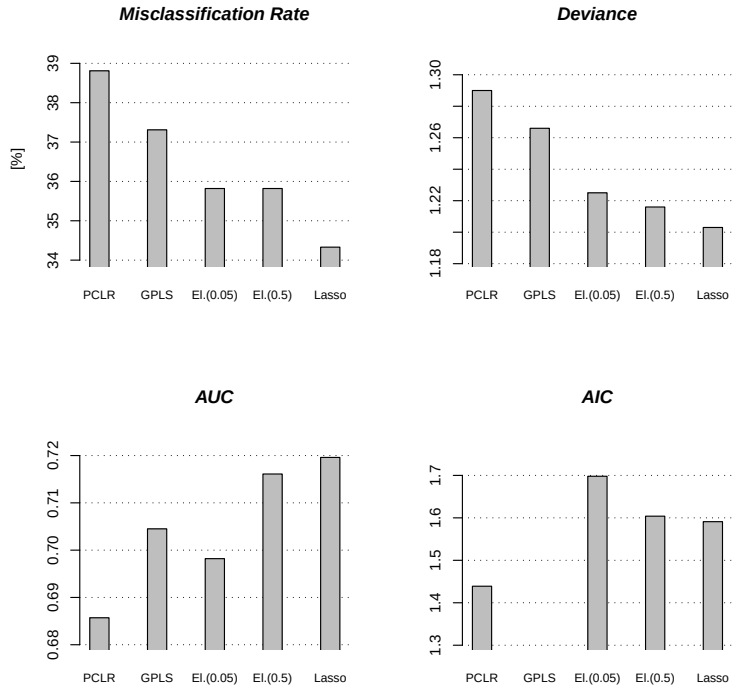


Figure 5.19: Test Results for Scenario 2.

5.3.3 Summary

In this chapter we studied the use of biased estimation methods to fit higher-dimensional marketing data. We considered two simulation scenarios. In the first scenario a sample of size 10000 was drawn and in the second scenario a sample of size 200. Each sample was split into a training set for parameter tuning and test set for model assessment. If we compare the training steps of the two test scenarios, it becomes apparent that the optimal parameters (λ values, PCLR and GPLS components) depend heavily on the size of the training sample. It seems that in $p > n$ cases (Scenario 2), more regularization is necessary to achieve good prediction performances. Furthermore, the plotted ROC curves provided us with meaningful results concerning the qualitative performance and pointed to a very good predictive ability of *all* proposed models. From the latter we can also infer that the training sample is a condign representative of the entire population (see stratified sampling), and moreover, that the training and test sample may be considered to be relatively homogeneous to each other. To assess ranking ability we scored the models using the "area under ROC curve" (AUC), deviance, misclassification rate and the Akaike information criterion (AIC). We discussed cost curves and incorporated unequal losses. Hence, showing that by varying the cutoff levels appropriate, misclassification cost can be reduced significantly. Concluding we can say that all models achieved comparatively good results with lasso standing on the top. However, we should keep in mind the fact that it is most likely that there exist elastic net models with α values smaller than one, performing equally well than lasso.

Chapter 6

Conclusion

A conclusion is the place where you get tired of thinking.[1]

In this thesis we discussed three different dimension reduction approaches to fitting logistic regression models for classification purposes. First of all, Principal Component Logistic Regression (PCLR), where we proposed two different methods based on cross validation, namely R-statistic and PRESS-MSEP, and the average eigenvalue rule for determining the significant set of principal components. As was only to be expected, there exist no ultimate rule for choosing the optimal number of pc's, and it is suggested to take three or at least two criteria into consideration. The next method introduced was General Partial Least Squares (GPLS). The derivation of the components of partial least squares is relatively straightforward in the linear setting, however, substantial algorithmic modifications are required in the general linear setting. Moreover, employing the cross validation procedure to GPLS to seek for the important components is from a computational point extremely costly, and it is not recommended to use this approach in case of huge data sets. The third approach is referred to as elastic net and includes ridge (L2 penalty) and lasso (L1 penalty) regression. The ability of lasso to shrink and select the features simultaneously shows superior performance compared to PCLR and GPLS. Linear combinations of the two penalties (elastic net $\alpha = 0.5/0.05$) did not lead to any further improvement in terms of prediction performance. However, determining the proper value for the regularization parameter is one of the most relevant problems of elastic net and can be troublesome or demand heavy computation (cross validation). We have demonstrated that all proposed techniques are powerful methods (if properly used) for feature selection in classification problems. Before concluding this chapter, we should also discuss the main points of criticism. These are considered to be the lack of transparency and accessibility, difficulties in understanding some of the concepts and problems in interpreting

[1] Arthur Bloch

the results, especially for researches with little mathematical background. This somewhat limits the utility in practice.

Appendix A

Algorithms and Methods

Appendix A gives on overview about certain methods and algorithms, but does not contain any claim to completeness. For any further detailed information please check the reference page.

A.1 Newton Rapson Method

Beginning with an initial guess for the solution, the Newton method uses the first terms of the Taylor polynomial evaluated at the initial guess to find another estimate being closer to the solution and continuing this process until convergence. Recall that the Taylor polynomial of degree n for a function f at the point x_0 may be written as (assuming that the first n derivatives of f at x_0 all exist)

$$\sum_{i=0}^{n} \frac{f^{(i)}(x_0)}{i!}(x - x_0)^i. \tag{A.1}$$

The first step in Newton's method is to take the first degree Taylor polynomial as an approximation for f and set it equal to zero.

$$f(x_0) + f^{'}(x_0)(x - x_0) = 0 \tag{A.2}$$

Solving for x, we get

$$x = x_0 - \frac{f(x_0)}{f^{'}(x_0)}. \tag{A.3}$$

Let $x_1 = x$ and continue in the same manner for obtaining $x_2, x_3, \cdots$ until successive approximations converge.

A.2 Predictor-Corrector Method

A.2.1 Predictor step

A linear approximation of $\beta(\lambda_{k+1})$ is given by

$$\widehat{\beta}^{k+} = \widehat{\beta}^{k} + (\lambda_{k+1} - \lambda_k)\frac{\partial \beta(\lambda)}{\partial \lambda} \tag{A.4}$$

where $\frac{\partial \beta(\lambda)}{\partial \lambda}$ may be computed from $\frac{\partial H(\beta(\lambda),\lambda)}{\partial \lambda} = \frac{\partial H}{\partial \lambda} + \frac{\partial H}{\partial \beta}\frac{\partial \beta}{\partial \lambda} = 0$, like the following

$$\frac{\partial \beta(\lambda)}{\partial \lambda} = -\left(\frac{\partial H}{\partial \beta}\right)^{-1}\frac{\partial H}{\partial \lambda} = (\mathbf{X}_A'\mathbf{V}_k\mathbf{X}_A)^{-1}\,\mathrm{sgn}\begin{pmatrix}0\\ \widehat{\beta}^k\end{pmatrix}, \tag{A.5}$$

where $\mathbf{V}_k = diag\{\frac{\partial G}{\partial \beta}(\mathbf{X}\widehat{\beta}^k)\}$ and $\mathbf{X}_A$ denotes the columns of X for the covariates in the current active set. It can be shown, that for a small step length $h = \Delta_k$ the approximated solution $\widehat{\beta}^{k+}$ differs from the real solution $\widehat{\beta}^{k+1}$ by $\mathcal{O}(h^2)$.

Assume that $\beta(\lambda)$ is a continuously differentiable function with respect to $\lambda \in (\lambda_{k+1}, \lambda_k]$, $\widehat{\beta}^{k+}$ is the linear approximate of $\widehat{\beta}^{k+1}$ as defined in (A.4) and $h = (\lambda_{k+1} - \lambda_k)$ is the step length. With the aid of the Taylor approximation

$$\widehat{\beta}^{k+1} = \widehat{\beta}^{k} + h\frac{\partial \beta}{\partial \lambda}\bigg|_{\lambda_k} + \mathcal{O}(h^2) = \widehat{\beta}^{k+} + \mathcal{O}(h^2). \tag{A.6}$$

it can easily be seen that $\widehat{\beta}^{k+}$ differs from the real solution $\widehat{\beta}^{k+1}$ by $\mathcal{O}(h^2)$.

A.2.2 Active step

After each corrector step the active set $\mathcal{A}$, starting with the intercept, has to be checked whether $\mathcal{A}$ should have been augmented. In [54] it was suggested to take as test procedure

$$\left|\mathbf{x}_j'(\mathbf{y} - G(\mathbf{X}\widehat{\beta}))\right| > \lambda \qquad \text{for any } j \in \mathcal{A}^c \Rightarrow \mathcal{A} \leftarrow \mathcal{A} \cup j \tag{A.7}$$

which may be derived from the Karush-Kuhn-Tucker (KKT) optimality conditions [52, p. 959]. The corrector step with the modified active set is repeated until condition A.7 suggests no further augmentation of the active set. Finally the the variables with zero coefficients are removed from the active set.

Minimizing expression (3.65) is equivalent to minimizing

$$-l(\beta) + \lambda\sum_j(\beta_j^+ + \beta_j^-) - \sum_j \lambda_j^+\beta_j^+ - \sum_j \lambda_j^-\beta_j^- \tag{A.8}$$

where $\beta = \beta^+ + \beta^-$, $\beta_j^+ \geq 0, \beta_j^- \geq 0$, for all j, i.e., splitting β in positive and negative coefficients, and $\lambda, \lambda_j^+, \lambda_j^+ \geq 0$ are the corresponding Lagrangian multipliers. Setting up the KKT conditions to this problem

$$\begin{aligned} -\mathbf{x}_j^{'}(\mathbf{y} - G(\mathbf{X}\widehat{\beta})) + \lambda - \lambda_j^+ = 0 \\ +\mathbf{x}_j^{'}(\mathbf{y} - G(\mathbf{X}\widehat{\beta})) + \lambda - \lambda_j^- = 0 \\ \lambda_j^+ \widehat{\beta}_j^+ = 0, \lambda_j^- \widehat{\beta}_j^- = 0 \quad \forall j = 1, \cdots, p \end{aligned} \tag{A.9}$$

imply that [24, p. 98]

$$\left| \mathbf{x}_j^{'}(\mathbf{y} - G(\mathbf{X}\widehat{\beta})) \right| < \lambda \Rightarrow \widehat{\beta}_j = 0 \quad \text{for } j = 1, \cdots, p$$

from which expression (A.7) can be derived.

A.3 Forward Stepwise Regression

Algorithm A.1: Forward Stepwise Regression.

1: Find predictor X_i most correlated with response variable Y
2: Build linear regression equation $\widehat{Y} = f(X_i)$ and test significance
3: **for** stopping criteria not met **do** ▷ e.g. When no predictor can be removed and the next best variable candidate cannot hold it's place in the equation
4: Choose X_j with highest partial F-Value (or AIC score,...) out of the set of predictors not in regression
5: Fit $\widehat{Y} = f(X_1, \cdots, X_j)$ and examine significance, F-Values and R^2 of regression
6: **if** partial F - Value of any predictor currently in the equation $\leq$ certain threshold α **then** ▷ e.g. smaller than the $90th$ or $95th$ percentile of the F - distribution
7: Remove this predictor from current equation
8: **end if**
9: **end for**

A.4 Iteratively Reweighted Partial Least Squares (IRPLS)

Algorithm A.2: IRPLS.

$\mathbf{E}_0 \leftarrow$ scaled $\mathbf{X}$; $\mathbf{z}_0 \leftarrow \psi(\mathbf{y})$; $\mathbf{V} \leftarrow \{G^{'}[\psi(\mathbf{y})]^2/\text{Var}[\mathbf{y}]\}$ ▷ Initialization
for until $\Delta\hat{\beta}$ small **do**
  for k=1 to R **do** ▷ R = rank($\mathbf{X}^{'}\mathbf{V}\mathbf{X}$)
    $\mathbf{w}_k \leftarrow (\mathbf{z}_{k-1}^{'}\mathbf{V}\mathbf{E}_{k-1}\mathbf{E}_{k-1}^{'}\mathbf{V}\mathbf{z}_{k-1})^{0.5}\mathbf{E}_{k-1}^{'}\mathbf{V}\mathbf{z}_{k-1}$ ▷ unit length orthog. loadings
    $\mathbf{t}_k = \mathbf{E}_{k-1}\mathbf{w}_k$ ▷ latent variables
    $\mathbf{t}_k \leftarrow center(wt : \mathbf{V})$; $||\mathbf{t}_k|| \rightarrow 1$ ▷ center with weighted mean + normalize
    $q_k \leftarrow (\mathbf{t}_k^{'}\mathbf{V}\mathbf{t}_k)^{-1}\mathbf{t}_k^{'}\mathbf{V}\mathbf{z}_{k-1}$ ▷ weighted lsfit
    $\mathbf{z}_k \leftarrow \mathbf{z}_{k-1} - \mathbf{t}_k q_k$ ▷ Update adjusted response
    $\mathbf{p}_k \leftarrow (\mathbf{t}_k^{'}\mathbf{V}\mathbf{t}_k)^{-1}\mathbf{t}_k^{'}\mathbf{V}\mathbf{E}_{k-1}$ ▷ weighted lsfit
    $\mathbf{E}_k \leftarrow \mathbf{E}_{k-1} - \mathbf{t}_k\mathbf{p}_k$ ▷ Update residual matrix
  end for
  $\mathbf{X}\hat{\beta} \leftarrow \text{intercept}(= wt.mean(\mathbf{z}_0)\ wt : \mathbf{V}) + \sum_{i=1}^{R} q_i\mathbf{t}_i$
  $\mathbf{V} \leftarrow G'[\mathbf{X}\hat{\beta}]/\text{Var}[\mathbf{y}]$
  $\mathbf{z}_0 \leftarrow \mathbf{X}\hat{\beta} + \text{diag}\{G'[\mathbf{X}\hat{\beta}]^{-1}\}(\mathbf{y} - G[\mathbf{X}\hat{\beta}])$
  $\mathbf{E}_0 \leftarrow \mathbf{X}$; $\mathbf{X} \leftarrow center(wt.\mathbf{V})$; $||\mathbf{X}|| \rightarrow 1$
end for
$\text{glm}(\mathbf{y} \sim (\mathbf{t}_1, \cdots, \mathbf{t}_s)\ s \leq R$ ▷ general least squares fit

A.5 Principal Component LR - Cross Validation

Algorithm A.3: PCA-CV.

1: $cv.fold \leftarrow$ fold-number; $max.comp \leftarrow$ number max. components; ▷ Initialization
2: **for** $k = 1$ to $cv.fold$ **do**
3: $\mathbf{X}^{test} \leftarrow \mathbf{X}_{\{k\}}$
4: $\mathbf{y}^{test} \leftarrow \mathbf{y}_{\{k\}}$
5: $\mathbf{X}^{train} \leftarrow \mathbf{X}/\mathbf{X}_{\{k\}}$
6: $\mathbf{y}^{train} \leftarrow \mathbf{y}/\mathbf{y}_{\{k\}}$
7: $(\mathbf{Z}^{train}, \mathbf{A}^{train}) \leftarrow \text{PCA}(\mathbf{X}^{train})$ ▷ Calc. score- and loading matrix
8: $\mathbf{Z}^{test} \leftarrow (\mathbf{X}^{test} - \mu^{train})\mathbf{A}^{train}$ ▷ Calc. score matrix test set
9: **for** $comp = 1$ to $max.comp$ **do**
10: $\hat{\mathbf{X}}^{test} \leftarrow \mathbf{Z}^{test}[, 1 : comp](\mathbf{A}^{train}[, 1 : comp])' + \mu^{train}$
11: $\text{PRESS}(comp) \leftarrow \sum_{i,j}(\hat{x}_{ij}^{test} - x_{ij}^{test})^2$
12: $\text{glmfit} \leftarrow \text{glm}(\mathbf{y}^{train} \sim \mathbf{Z}^{train}[, 1 : comp])$ ▷ general ls fit
13: $\hat{\mathbf{y}}^{test} \leftarrow \text{glmfit}(\mathbf{Z}^{test}[, 1 : comp])$ ▷ Prediction using test set
14: $\text{PRESS-MSEP}(comp) \leftarrow \sum_i(\hat{y}_i^{test} - y_i^{test})^2$
15: **end for**
16: **end for**
17: **return** (PRESS, PRESS-MSEP)

A.6 Stratification

Algorithm A.4: Stratification.

1: $train.sample \leftarrow NULL$; $obs \leftarrow NULL$; $frac \leftarrow$ Fraction of responders of total population ▷ Initialization
2: **for** $train.sample \leq train.size$ **do**
3: $obs \leftarrow$ Draw an observation at random without replacement from the population
4: **if** obs is in stratum of responders **then**
5: Keep obs with probability $frac$ and if so add obs to $train.sample$
6: **else** ▷ obs is in stratum of non-responders
7: Keep obs with prob. $(1 - frac)$ and if so add obs to $train.sample$
8: **end if**
9: **end for**
10: **return** ($train.sample$)

Appendix B

Simulations (Continued)

Appendix B mainly contains figures of the simulation results of the second test scenario, which are tried to be consistent with the figures produced in the first scenario.

B.1 Figures of Test Scenario 2

Parameter Tuning

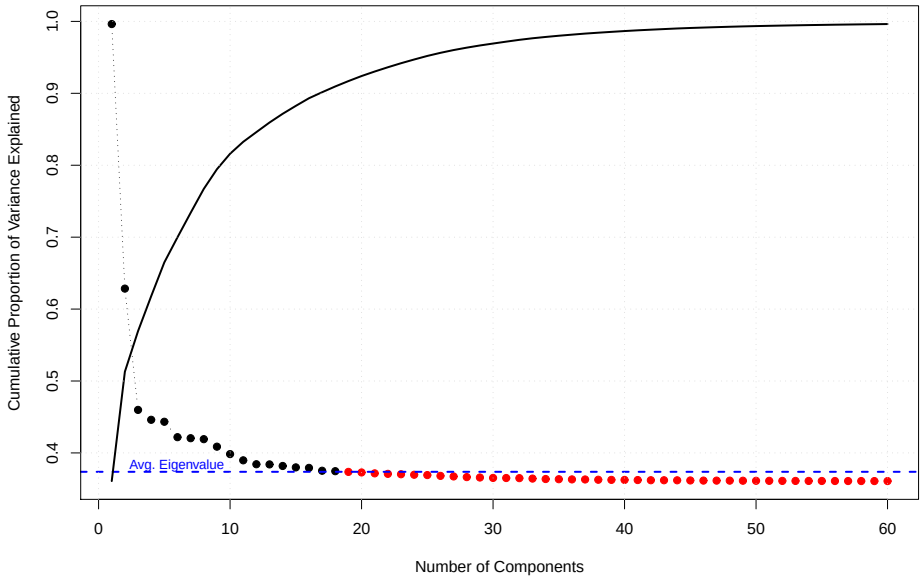


Figure B.1: Eigenvalues of the sample training covariance matrix.

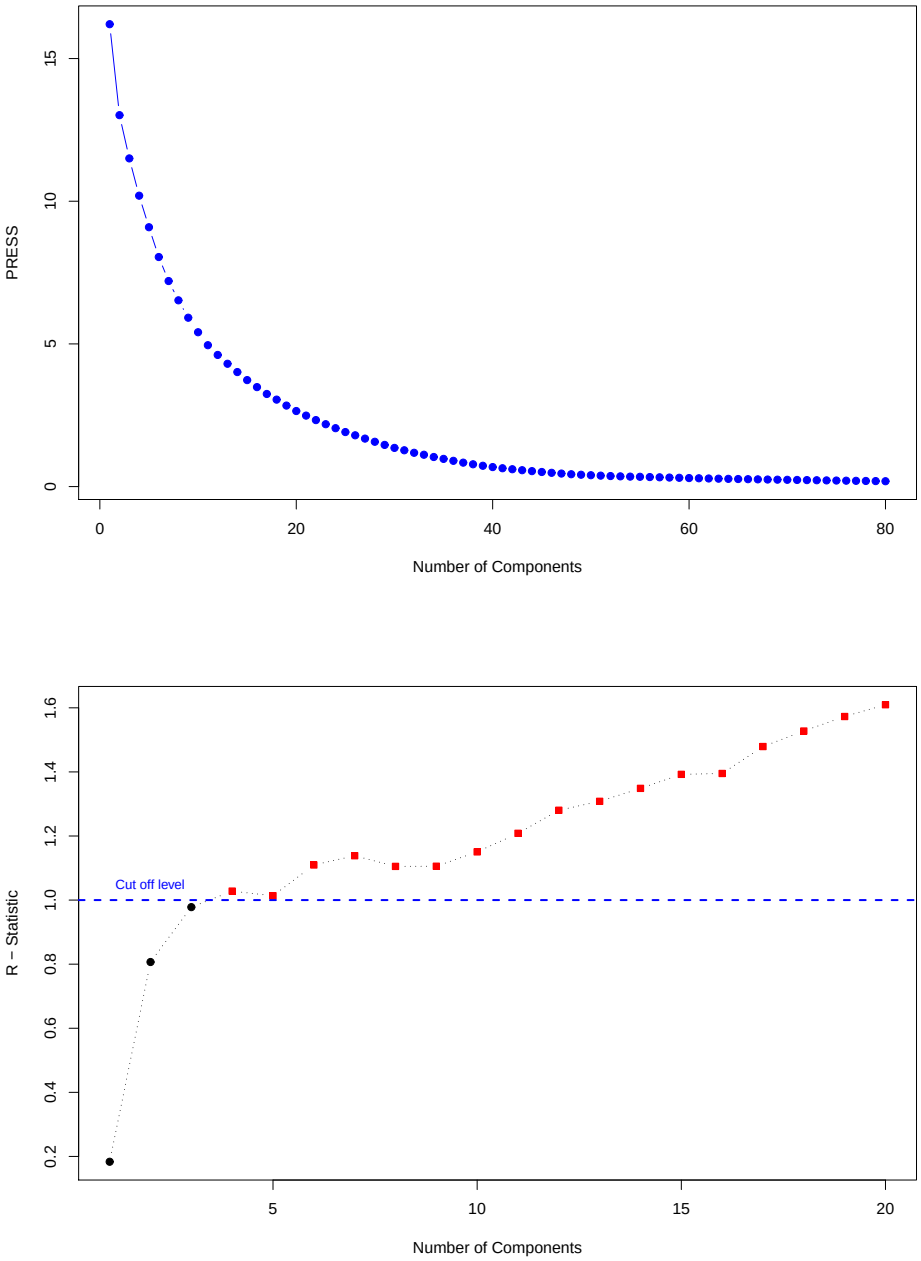


Figure B.2: (top) PRESS evaluated by 10 fold cross validation and averaged over 30 consecutive runs; (bottom) R - Statistic.

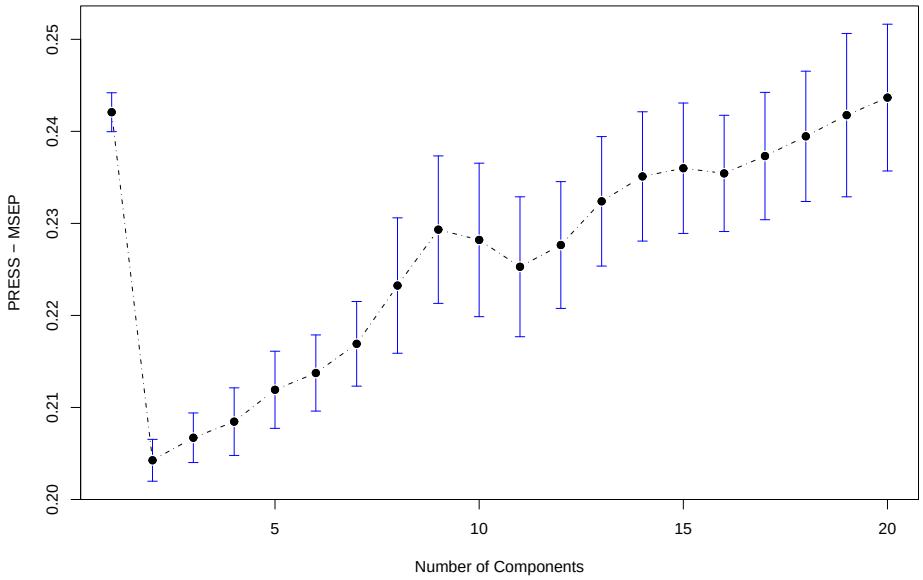


Figure B.3: PRESS-MSEP evaluated by 10 fold cross validation and averaged over 30 consecutive runs.

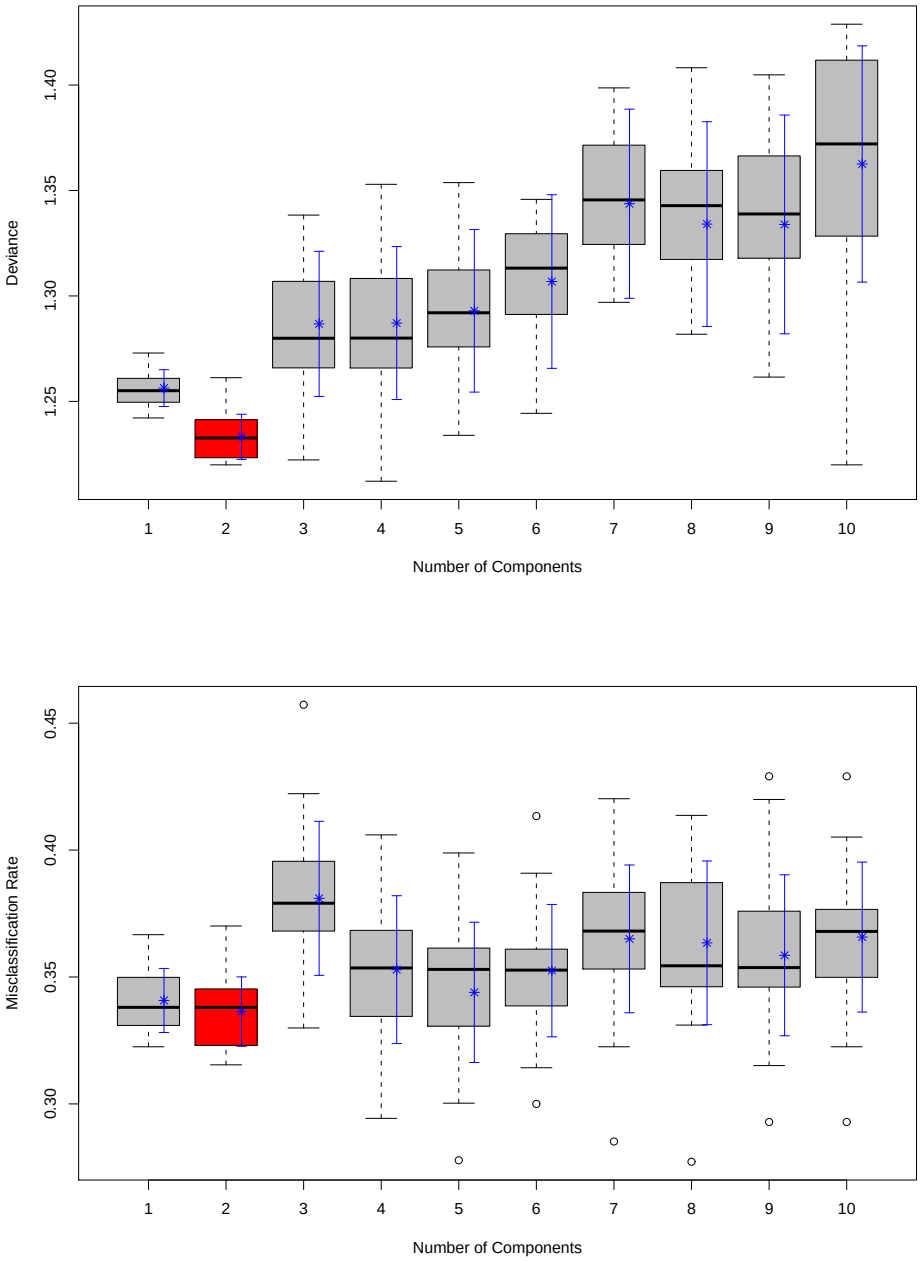


Figure B.4: (top) General partial least squares training tests: Boxplot of 5-fold CV estimates with binomial deviance loss function over 20 consecutive runs; (bottom) General partial least squares training tests: Boxplot of 5-fold CV estimates with 0-1 loss function over 20 consecutive runs.

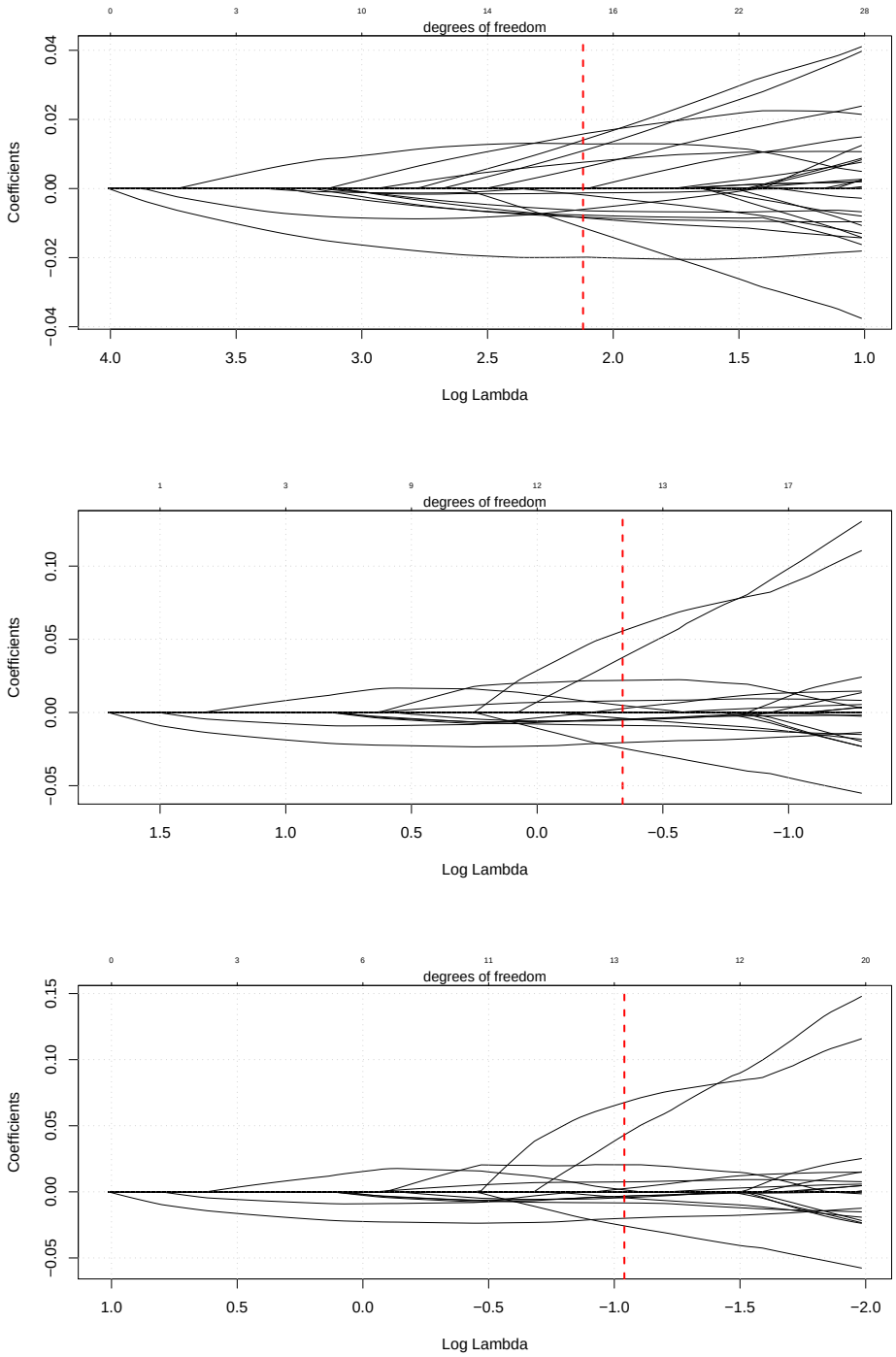


Figure B.5: Coefficient profile plot of the coefficient paths (training set): (top) Elastic net $\alpha = 0.05$; (middle) Elastic net $\alpha = 0.5$; (bottom) Lasso (elastic net $\alpha = 1$).

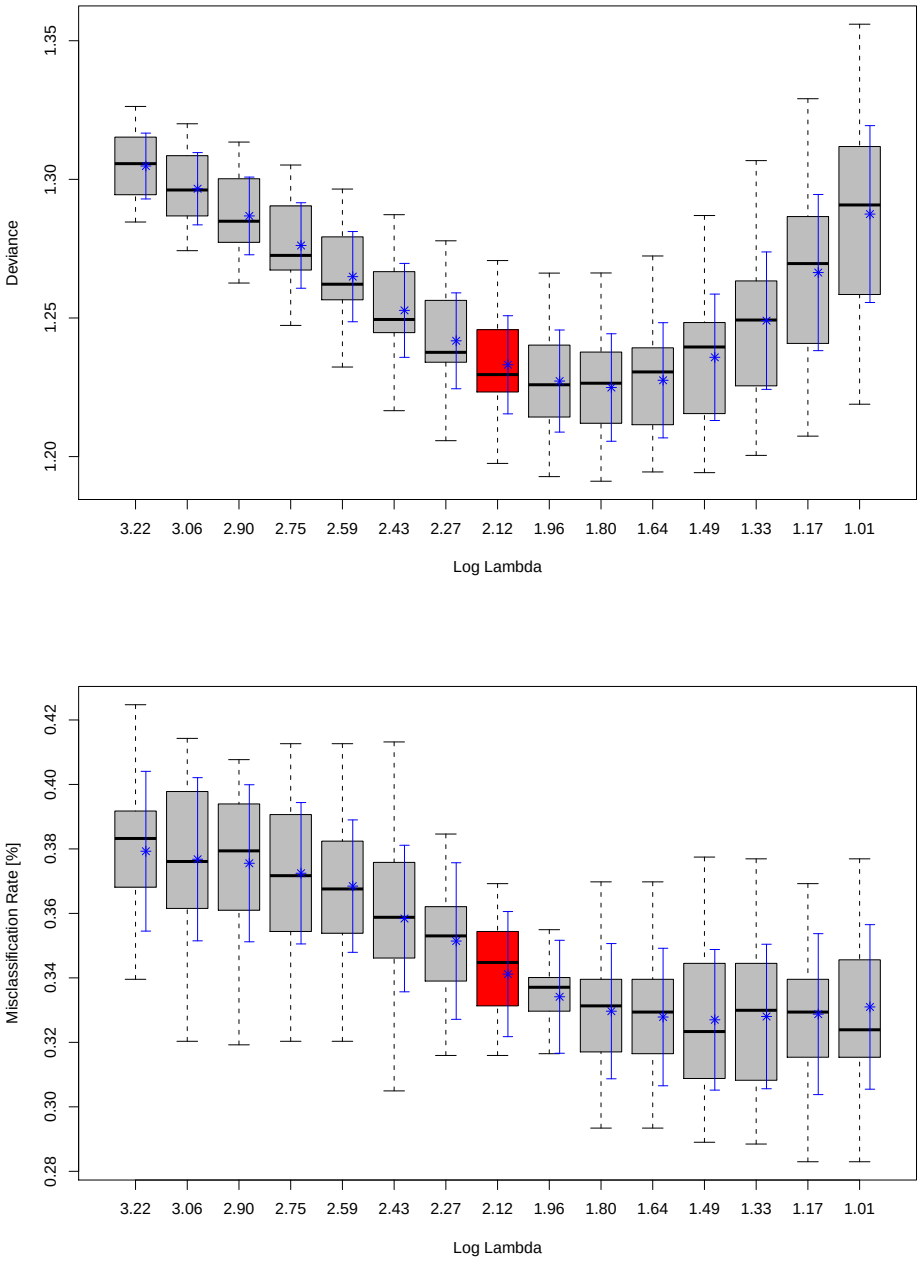


Figure B.6: Elastic net $\alpha = 0.05$ training tests: (top) Boxplot of 10-fold CV estimates with binomial deviance loss function over 30 consecutive runs; (bottom) Boxplot of 10-fold CV estimates with 0–1 loss function over 30 consecutive runs.

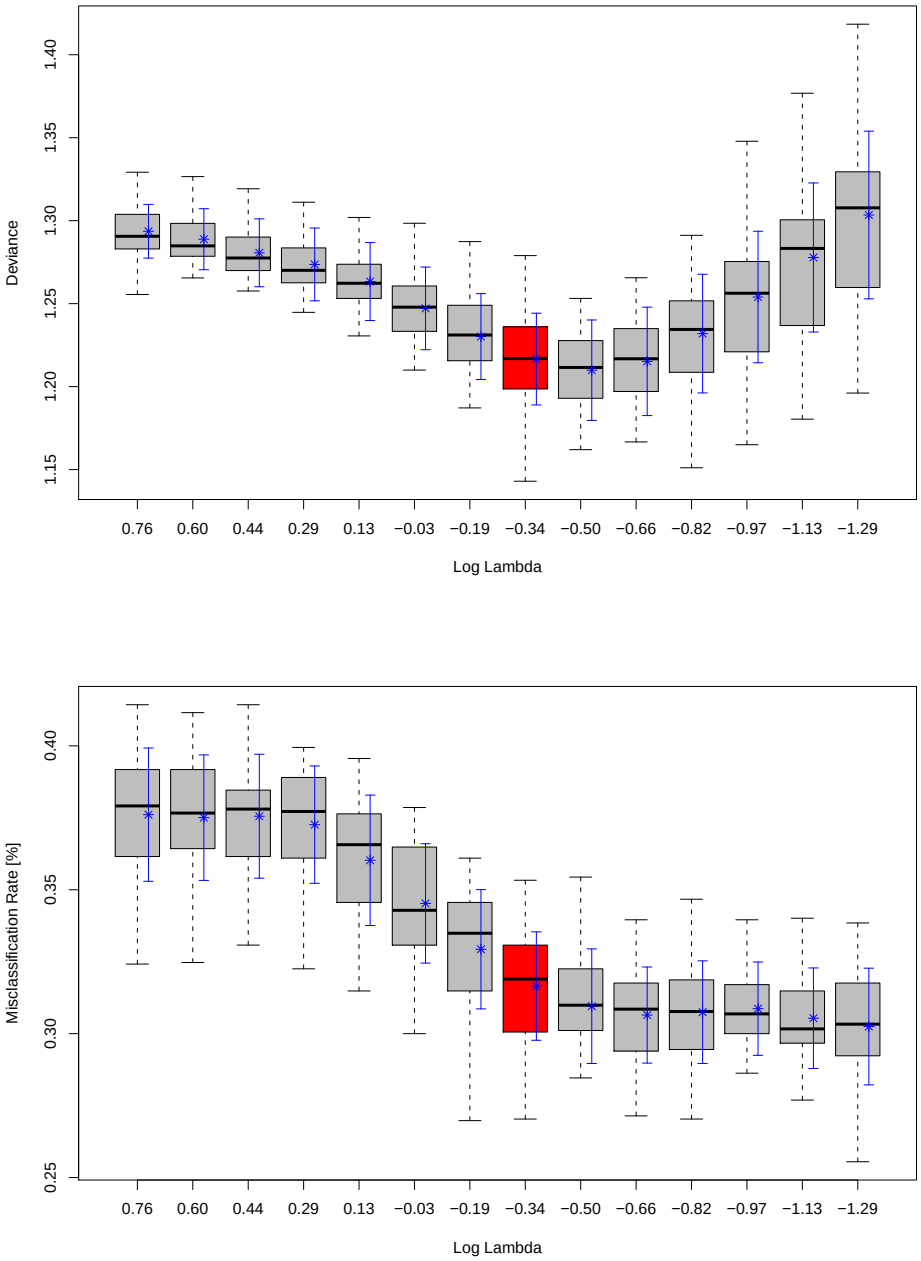


Figure B.7: Elastic net $\alpha = 0.5$ training tests: (top) Boxplot of 10-fold CV estimates with binomial deviance loss function over 30 consecutive runs; (bottom) Boxplot of 10-fold CV estimates with 0–1 loss function over 30 consecutive runs.

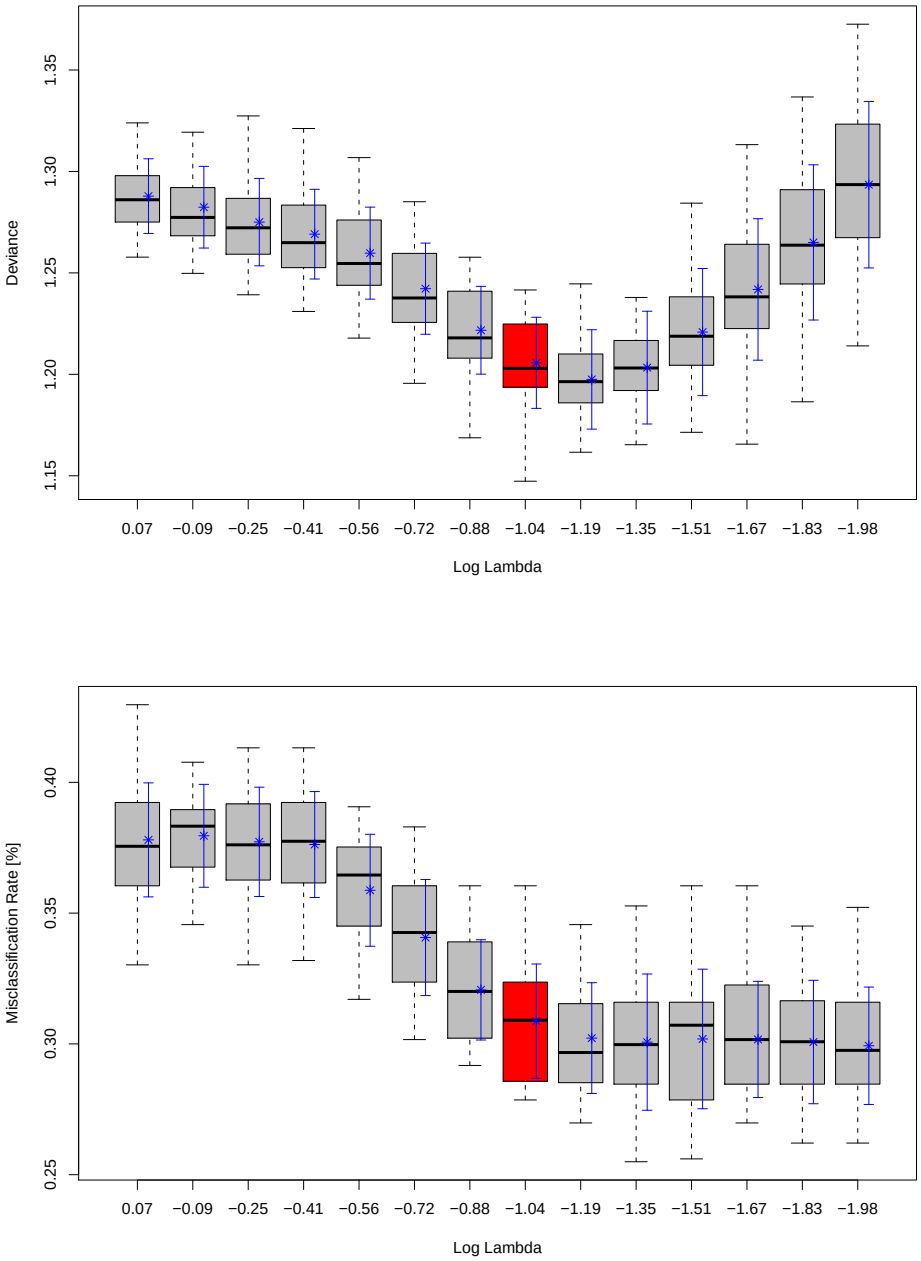


Figure B.8: Elastic net $\alpha = 1$ training tests: (top) Boxplot of 10-fold CV estimates with binomial deviance loss function over 30 consecutive runs; (bottom) Boxplot of 10-fold CV estimates with 0–1 loss function over 30 consecutive runs.

Testing

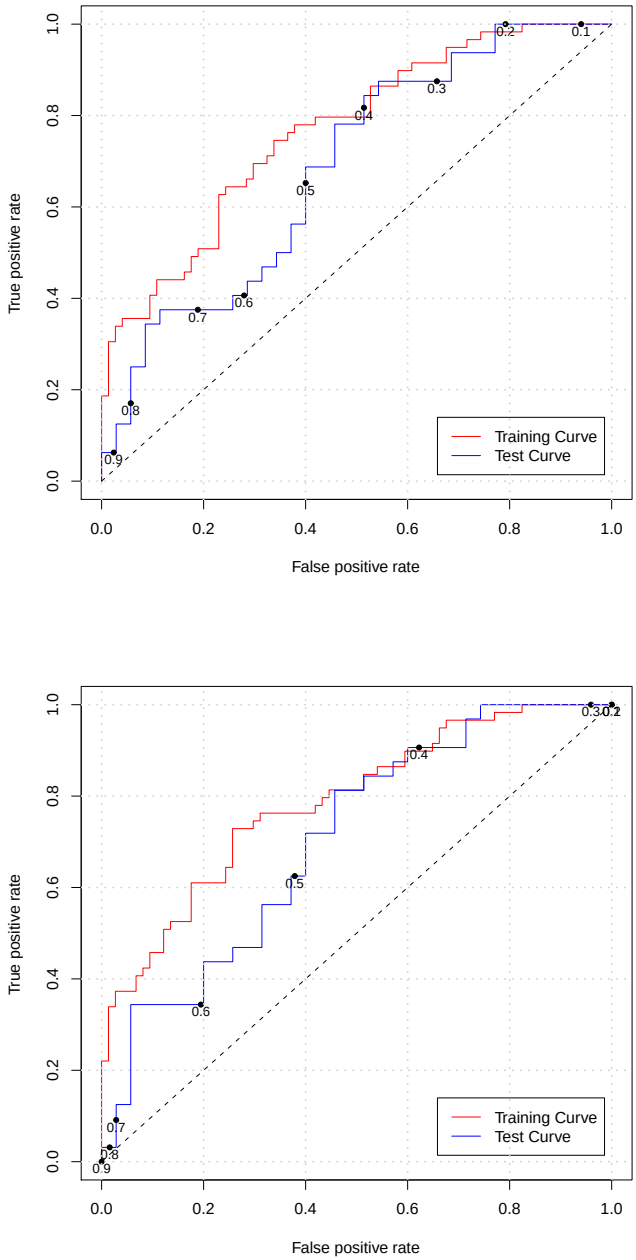


Figure B.9: ROC training and test curves plus cutoff levels: (top) Principal Component Logistic Regression (PCLR); (bottom) General Partial Least Squares (GPLS).

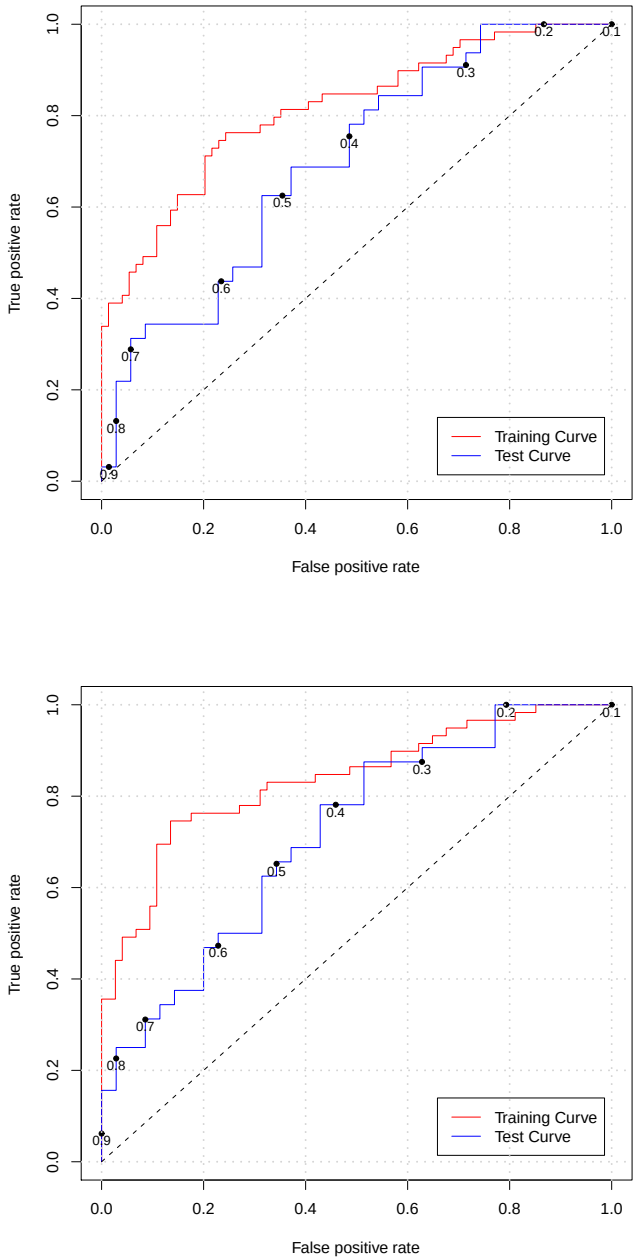


Figure B.10: ROC training and test curves plus cutoff levels: (top) Elastic net $\alpha = 0.05$; (bottom) Elastic net $\alpha = 0.5$.

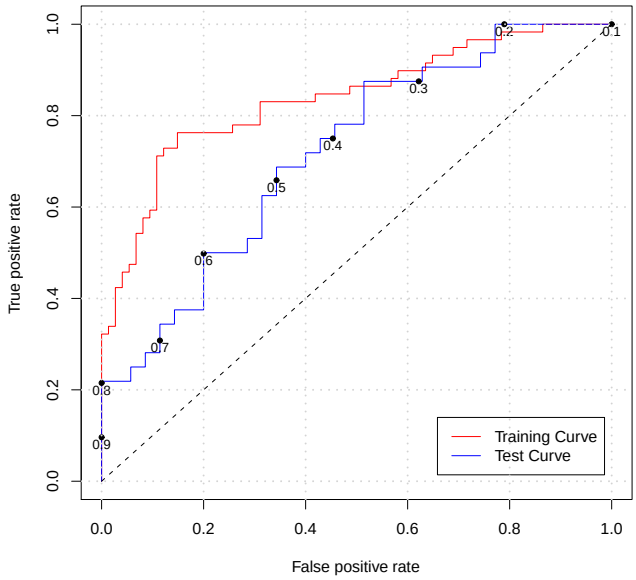


Figure B.11: Lasso (elastic net $\alpha = 1$): ROC training and test curves plus cutoff levels.

B.2 Miscellaneous

B.2.1 Scenario 1: Linear Combination of the 1st Principal Component

ortsgroesse	kaufkraftkl	adresseseit	BA_Lebzeit	BW_Lebzeit	vskunde_seit	smxbw02	smxbw07
-3.057491e-03	-1.015385e-03	4.642219e-04	4.642219e-04	4.410654e-03	4.642219e-04	-6.523387e-03	-3.821938e-02
smxbw08	smxbw09	datltztbest_vs	dlbvsakt01	dlbvsakt11	dlbvsakt12	dlbvsakt13	dlbvsakt14
-5.676676e-02	-5.676676e-02	-3.645743e-02	-8.680965e-02	-8.813133e-03	-1.929355e-02	-5.053309e-03	-2.688091e-05
dlbvsakt15	dlbvsakt16	dlbvsakt17	dlbvsakt18	dlbvsakt19	dlbvsakt26	dlbvsakt27	dlbvsaktspz
-9.697076e-03	-5.731993e-02	-9.235419e-03	-8.060835e-02	-1.580062e-02	-3.716416e-02	-9.953108e-03	-1.178826e-01
dlbvsaktlux	dlbvs04	dlbvs05	dlbvs07	dlbvs08	dlbvs37	dlbvdob	dlbvtextil
-1.253481e-02	-9.293923e-02	-1.372964e-01	-1.088260e-01	-1.190257e-01	-7.436322e-02	-1.344804e-01	-1.081510e-01
dlbvjung	dlbvalt	dlbvbig	ashp03	ass01	ass02	ass03	ass04
-1.531877e-01	-1.469667e-01	-9.348576e-02	4.956100e-03	1.066389e-02	1.656113e-02	1.041044e-02	6.875257e-03
ass05	ass06	ass07	ass08	ass09	asdob	ashob	assport
1.446233e-02	7.388546e-03	7.913962e-03	9.739431e-03	2.251825e-04	1.961697e-02	1.272801e-02	6.106641e-03
astextil	asjung	asalt	asbig	asindiv	asindiv2	bwhp1_3	bavs0
1.820246e-02	1.438002e-02	1.466989e-02	7.016506e-03	1.968459e-02	1.257013e-02	9.013318e-03	5.803014e-03
bavs1	bavs2	bavs3	bavs4	bwvs0	bwvs1	bwvs2	bwvs3
2.083600e-02	1.873044e-02	1.740546e-02	1.552158e-02	2.929308e-03	4.611599e-03	4.372544e-03	3.822845e-03
bwvs4	bavs0_1	bavs0_2	bavs0_3	bavs0_4	bavs0_5	bwvs0_1	bwvs0_2
3.478589e-03	2.663901e-02	4.536945e-02	6.277491e-02	7.829649e-02	3.956643e-02	4.870608e-03	5.353290e-03
bwvs0_3	bwvs0_4	bwvs0_5	bwvs0_6	bwvs0_7	vsum0	vsum1	vsum2
5.476919e-03	5.389876e-03	5.231081e-03	5.381912e-03	5.303917e-03	5.852286e-04	1.035568e-03	1.283460e-03
vsum0_4	rqs00	rqs03	rqs05	rqs06	rqs08	neukundenart	nkbw
1.424326e-03	2.220719e-01	3.274624e-01	3.973186e-01	3.725015e-01	3.303159e-01	-2.983557e-03	-1.172813e-03
nkbestellw	nkbw_dob	nkbw_technik	nkbw_alt	nkbw_dobalt	nkbw_big	nkbw_dobbig	nkbw_hp1
-2.285446e-03	4.721503e-04	-1.436693e-03	-1.383545e-04	1.669990e-04	1.647344e-04	2.145563e-04	3.394007e-04
bas0a01	bas0a09	bas0a16	bas0a18	bas1a01	bas1a09	bas1a16	bas1a18
2.936339e-03	6.897370e-04	2.306772e-04	3.422163e-04	7.701879e-03	3.193930e-03	8.003494e-04	1.316546e-03
bas2a01	bas2a09	bas2a16	bas2a18	bas3a01	bas3a09	bas3a16	bas3a18
6.457226e-03	1.898196e-03	6.086095e-04	1.200885e-03	6.641066e-03	1.688576e-03	7.072711e-04	9.979396e-04
bas4a01	bas4a09	bas4a16	bas4a18	bas0_1a01	bas0_1a09	bas0_1a16	bas0_1a18
6.058506e-03	1.669972e-03	6.522066e-04	4.707256e-04	1.063822e-02	3.883667e-03	1.031027e-03	1.658762e-03
bas0_1a26	bas0_2a01	bas0_2a09	bas0_2a16	bas0_2a26	bas0_3a01	bas0_3a09	bas0_3a18

4.080511e-04	1.709544e-02	5.781862e-03	1.639636e-03	7.006175e-04	2.373651e-02	7.470439e-03	3.857587e-03
bas0_3a26	bas0_4a01	bas0_4a02	bas0_4a06	bas0_4a09	bas0_4a16	bas0_4a18	bas0_4a26
9.179499e-04	2.979502e-02	7.146680e-04	2.825691e-03	9.140410e-03	2.999114e-03	4.328312e-03	1.159595e-03
bas0_5a09	bas0_6a09	bas0_7a09	bas0_8a09	bas0_9a09	bas0_10a09	bws0a01	bws0a16
5.092125e-03	6.780702e-03	8.450673e-03	3.586772e-03	5.256743e-03	3.358548e-03	3.029926e-03	1.841022e-03
bws1a01	bws1a16	bws2a01	bws2a16	bws3a01	bws3a16	bws4a01	bws4a16
4.286115e-03	2.814084e-03	3.950949e-03	1.924614e-03	3.670332e-03	2.279765e-03	2.858476e-03	2.134733e-03
bws0_1a01	bws0_1a16	bws0_1a26	bws0_2a01	bws0_2a16	bws0_2a26	bws0_3a01	bws0_3a16
4.704537e-03	3.118011e-03	1.778519e-03	5.104384e-03	3.149428e-03	2.219730e-03	5.241563e-03	3.324592e-03
bws0_4a01	bws0_4a07	bws0_4a16	bws0_4a17	bws0_4a18	bws0_10a01	bws0_10a16	bwenrs0a16
5.062464e-03	4.621644e-03	3.435996e-03	1.593176e-03	4.902329e-03	3.842637e-03	2.777203e-03	2.110821e-03
bwenrs1a16	bwenrs2a16	bwenrs3a16	bwenrs0_1a16	bwenrs0_1a24	bwenrs0_2a16	bwenrs0_2a26	bwenrs0_3a16
3.295019e-03	2.542089e-03	2.830886e-03	3.639110e-03	4.051948e-03	3.764590e-03	2.603576e-03	3.929585e-03
bwenrs0_3a24	bavs1_2	bavs1_3	bavs1_4	bavs1_5	bavs2_2	bavs3_2	bavs3_3
5.932330e-03	8.012504e-03	1.234007e-02	1.541975e-02	1.889480e-02	1.207257e-02	6.180200e-03	1.047350e-02
bavs3_4	bavs3_5	bavs3_6	bavs4_2	bavs4_3	bavs4_4	bavs5_2	bavs7_2
1.474996e-02	1.766792e-02	9.054608e-03	2.320947e-03	4.765247e-03	8.039448e-03	1.024187e-02	4.628759e-03
bavs7_3	bavs9_3	bavs9_4	bavs29_5	badob_2	badob_3	badob_4	badob_6
7.562278e-03	6.043293e-05	1.337267e-04	2.868918e-03	3.442574e-02	2.853250e-02	3.019223e-02	4.347217e-02
badob_7	badob_8	badob_9	bahob_1	bahob_2	bahob_3	bahob_4	bahob_5
7.200466e-02	1.021969e-01	1.262883e-01	1.987524e-03	9.575247e-03	1.053063e-02	8.662658e-03	8.842939e-03
bahob_6	bahob_7	bahob_8	bahob_9	bahob_10	bahob_11	bahob_12	bahob_13
1.156277e-02	2.209340e-02	3.075606e-02	3.959900e-02	2.010588e-02	2.876853e-02	3.761147e-02	1.919329e-02
bahob_14	bahob_15	bakob_1	bakob_2	bakob_3	bakob_4	bakob_5	bakob_6
2.803623e-02	1.750560e-02	9.937730e-04	3.790366e-03	4.707726e-03	3.495288e-03	4.160259e-03	4.784139e-03
bakob_7	bakob_8	bakob_9	bakob_10	bakob_11	bakob_12	bakob_13	bakob_14
9.491865e-03	1.298715e-02	1.714741e-02	8.498092e-03	1.199338e-02	1.615364e-02	8.203014e-03	1.236327e-02
bakob_15	batec_4	badaj_2	badaj_3	badaj_4	badaj_5	badaj_9	baspo_9
7.655548e-03	1.516969e-04	1.594965e-02	1.285555e-02	1.235753e-02	1.128864e-02	5.658678e-02	9.893218e-03
batex_2	batex_3	batex_4	batex_5	batex_6	batex_7	batex_8	batex_9
4.749625e-02	4.287227e-02	4.202513e-02	3.630339e-02	5.947519e-02	1.023475e-01	1.443726e-01	1.806760e-01
bwvs1_5	bwvs2_2	bwvs2_5	bwvs3_2	bwvs3_3	bwvs3_4	bwvs3_5	bwvs3_6
7.752882e-03	7.969712e-03	9.033786e-03	6.122327e-03	6.715985e-03	6.954477e-03	7.098410e-03	6.305032e-03
bwvs3_7	bwvs3_8	bwvs4_2	bwvs4_3	bwvs4_4	bwvs4_5	bwvs5_4	bwvs6_5
6.636742e-03	6.832818e-03	3.846313e-03	4.515058e-03	5.129543e-03	5.467054e-03	7.625831e-03	7.091079e-03
bwvs7_2	bwvs7_3	bwvs8_2	bwvs8_4	bwvs8_5	bwvs9_4	bwvs28_3	bwdob_6
5.182645e-03	5.797819e-03	4.145132e-03	6.435001e-03	6.530224e-03	7.344205e-04	2.584875e-03	9.974655e-03
bwdob_7	bwdob_8	bwdob_9	bwhob_6	bwhob_7	bwkob_6	bwkob_9	bwkob_15
1.023400e-02	1.040400e-02	1.024482e-02	5.667262e-03	6.646296e-03	3.176700e-03	3.382390e-03	2.782135e-03
bwtec_15	bwhtx_1	bwson_5	bwson_6	bwson_7	bwson_8	bwdaj_8	bwmst_6
-1.090276e-03	1.359535e-03	2.779100e-03	3.609468e-03	4.016534e-03	4.000359e-03	9.478541e-03	5.548807e-03
bwmst_7	bwmst_8	bwtfk_6	bwtfk_7	bwtfk_8	bwtex_6	bwtex_7	bwtex_8
6.672549e-03	7.007979e-03	-2.698959e-04	-4.256755e-04	-5.528323e-04	1.012085e-02	1.030126e-02	1.038874e-02
bwtex_9	bwntx_2	vsums4_2	vsums7_2	vsums4_3	vsums7_3	vsums1_4	vsums3_4
1.011119e-02	3.785499e-04	2.233547e-03	3.462249e-03	2.642301e-03	3.804877e-03	4.574302e-03	4.148098e-03
vsums4_4	vsums3_5	vsums3_6	vsumsodob_2	vsumsodob_3	vsumsodob_5	vsumsotec_6	vsumsospo_6
2.956406e-03	4.241543e-03	3.905543e-03	5.925771e-03	5.422329e-03	4.425672e-03	-1.141557e-03	1.017177e-03
vsumsodob_8	vsumsodob_9	dbws5s01	dbws5s03	dbws5s04	dbws5s24	dbws6s01	dbws6s03
6.689268e-03	6.601336e-03	6.455879e-03	5.820303e-03	3.866689e-03	-8.376052e-05	6.618576e-03	6.009587e-03
dbws6s04	dbws7s01	dbws7s03	dbws7s04	dbws10s07	dbws10s08	dbwsdob02	dbwsdob03
4.707682e-03	6.581515e-03	6.169873e-03	5.326063e-03	4.799423e-03	5.376299e-03	6.934170e-03	5.824953e-03
dbwsdob04	dbwsdob05	dbwsdob06	dbwsdob07	dbwsdob08	dbwsdob09	dbwskob05	dbwstec06
5.606261e-03	4.717913e-03	7.121010e-03	6.024886e-03	5.126665e-03	4.485849e-03	1.076405e-03	-1.063851e-03
dbwstec07	dbwsdaj06	dbwstex06	dbwstex07	askzs0	askzs1	askzs2	askzs3
-1.356933e-03	6.882827e-03	6.637537e-03	5.245344e-03	1.367076e-02	3.586637e-02	3.421513e-02	3.109441e-02
askzs4	askzs5	askzs6	askzs7	askzs8	bwdob14_01	bavsjgs0_1	bwvsdamen_9
2.824557e-02	4.077300e-02	5.185829e-02	5.624529e-02	5.689765e-02	9.619449e-03	1.264126e-02	1.014422e-02
bwvsherren_9	bwjungs34	dlbhob1	dlbhob	dlbvsaltsk	antbw5	bwsortalt_4	bwsorthob_4
6.826565e-03	6.007829e-03	-1.234286e-01	-1.283316e-01	-1.104300e-01	2.264064e-03	8.337286e-03	7.014336e-03
bwsorttxt_4	dbws234	dbws789	bwson2_7	bavm0_3	bavm0_4	bavm5_10	bavm0_2
1.036207e-02	4.326749e-03	2.984160e-03	4.053136e-03	3.370807e-02	3.730214e-02	2.073375e-02	3.028270e-02
bwvm0_3	bwvm0_4	bwvm0_2	bwa18	bwa07	bwa17	pbwwas08	lba24_9
4.240564e-03	4.467496e-03	4.053057e-03	4.902329e-03	4.621644e-03	1.593176e-03	1.927845e-03	-9.570050e-02
bwdob2_6	pbwsdob09	pbwstex08	bwhaustex	ass0_3s34	ba2hpkz167	bw4skz3489	ba4hpkz127
9.619449e-03	5.835650e-03	5.699994e-03	4.053136e-03	1.063391e-02	1.456024e-02	8.337286e-03	4.720232e-02
dbw2skz234	dlbvdaj	anrede_code	age_1	age_2	age_3	age_4	age_5
7.175426e-03	-1.583034e-01	1.047477e-03	-1.309251e-04	2.953192e-04	5.051845e-04	3.849514e-05	-7.059489e-04

B.2.2 Scenario 1: Covariates of elastic net $\alpha = 0.5$

[1] "ortsgroesse"	"smxbw02"	"smxbw07"	"smxbw08"	"smxbw09"	"datltztbest_vs"	"dlbvsakt01"
[8] "dlbvsakt11"	"dlbvsakt12"	"dlbvsakt13"	"dlbvsakt15"	"dlbvsakt16"	"dlbvsakt17"	"dlbvsakt19"
[15] "dlbvsakt26"	"dlbvsakt27"	"dlbvsaktspz"	"dlbvs04"	"dlbvs05"	"dlbvs08"	"dlbvs37"
[22] "dlbvdob"	"dlbvtextil"	"dlbvjung"	"bavs3"	"bavs0_1"	"rqs00"	"rqs03"
[29] "rqs05"	"rqs06"	"rqs08"	"neukundenart"	"nkbw_alt"	"bas4a01"	"bas0_4a01"
[36] "bas0_4a02"	"bas0_9a09"	"bavs1_5"	"bavs3_5"	"bavs29_5"	"badob_9"	"bakob_12"
[43] "badaj_2"	"badaj_9"	"baspo_9"	"batex_3"	"batex_6"	"batex_9"	"bwtfk_8"
[50] "dbws5s03"	"dbwsdaj06"	"askzs2"	"askzs4"	"askzs6"	"askzs7"	"askzs8"
[57] "dlbhob1"	"dlbvsaltsk"	"dbws234"	"bavm0_4"	"lba24_9"	"dlbvdaj"	

B.2.3 Scenario 2: Covariates of elastic net $\alpha = 0.5$ and $\alpha = 1$

```
[1] "smxbw02"     "dlbvsakt16"  "dlbvsakt18"  "dlbvsakt26"  "dlbvsaktspz" "dlbvs37"     "dlbvtextil"  "dlbvjung"
[9] "bavs0_4"     "rqs08"       "batex_9"     "askzs6"      "askzs8"
```

Appendix C

Source Code

Appendix C includes the R - Source code divided into a main file and a function file. In case of difficulties in understanding or ideas for improvements one should not hesitate to contact me.

C.1 main.r

```
#*****************************************************************************
# Main File
#-----------------------------------------------------------------------------
#Author: Grosswindhager Stefan
#Last Update: 26.09.09
#Title: Dimension reduction techniques for micro array data
#*****************************************************************************
   library(glmnet)                          #load libraries
   library(gpls)
   library(pls)
   library(ROCR)
   library(gregmisc)
   source("C:\\USERS\\STEFAN\\Diplomarbeit\\R\\function.r")   #load the function r file
   path = 'C:\\USERS\\STEFAN\\Desktop\\'                #Specify path of dataset
   filename = "DBAB3135_VTK_VAR3_0002.csv"              #Specify filename of dataset
   dataset <- init_data(path,filename)                  #Read the data
#-----------------------------------------------------------------------------
   #Draw random sample
   sample_size=10000  #Scenario 1      # sample_size=200  #Scenario 2
   dataset<-dataset[sample(1:nrow(dataset),sample_size,replace=FALSE),]

#-----------------------------------------------------------------------------
# Dataset manipulation

  #categorize variable: alter1
  index<-1:length(dataset$alter1)
  age_1=age_2=age_3=age_4=age_5<-rep(0,length(dataset$alter1))
  qu<-quantile(dataset$alter1, probs = c(0, 0.2, 0.4, 0.6, 0.8, 1))
  age_cut<-(cut(dataset$alter1, breaks=qu,right = FALSE,left=TRUE,labels=FALSE))
  age_1[index[age_cut==1]]=1
  age_2[index[age_cut==2]]=1
  age_3[index[age_cut==3]]=1
  age_4[index[age_cut==4]]=1
  age_5[index[age_cut==5]]=1

  #Delete variables which should not be in the regression model
  dataset$plzl = dataset$nkumsant = dataset$lvsreakt = dataset$alter1=NULL
  dataset<- data.frame(cbind(dataset, age_1,age_2,age_3,age_4,age_5))
  X <- data.matrix(dataset[,2:ncol(dataset)])   #Get the design matrix. In the first column is the response variable
  y <- dataset$REAGIERER    #y <- response variable
  rownames(X)<-1:nrow(X)
  names(y)<-1:length(y)
  del <- NULL
  X_temp <- X
  X_temp <- scale(X_temp,scale=FALSE)   # center the matrix
  sum <- colSums(abs(X_temp))           # calculate columns sums
  index<-1:ncol(X)
```

```
  del <- index[sum==0]                  #check if i have zero columns
  X <- subset(X, select = -del)         #remove zero columns
  X_temp<-X

  #filter categorical covariates
  X_temp<-X_temp%%1
  sum <- colSums(abs(X_temp))
  index<-1:ncol(X_temp)
  categorical <- index[sum==0]
  #scale continuous variables
  X[,-categorical]<-scale(X[,-categorical],center=FALSE,scale=TRUE)
  X[,3]<-scale(X[,3],center=FALSE,scale=TRUE) # variable adresseseit:: conceptually continuous
  X[,4]<-scale(X[,3],center=FALSE,scale=TRUE) # variable BA_Lebzeit: conceptually continuous
  X[,6]<-scale(X[,3],center=FALSE,scale=TRUE) # variable vskunde_seit: conceptually continuous
  X_temp=NULL

 #-------------------------------------------------------------------------------
 #Stratification
  train_size <- 6666  #Scenario 1   #train_size <- 133  #Scenario 2
  frac<-sum(y)/length(y)     #fraction of responders
  train_sample <- strat(y,trains_size,frac)
  frac_temp<-sum(y[train_sample])/train_size

#-------------------------------------------------------------------------------
#Elastic net training
#-------------------------------------------------------------------------------
  runs <- 30     #number of runs
  n_lambda<-20   #number of max lambda values
  cv_pred_dev<-matrix(nrow = n_lambda, ncol = runs)
  cv_pred_cr<-matrix(nrow = n_lambda, ncol = runs)
  for (i in 1:runs) {
     elastic_output<-elastic_net_train(X[train_sample,],y[train_sample],n_lambda,FALSE,1,10)
     cv_pred_dev[1:length(elastic_output[[1]]),i]<-elastic_output[[1]]
     cv_pred_cr[1:length(elastic_output[[2]]),i]<-elastic_output[[2]]
  }
  lambda_seq<-elastic_output[[3]]
#-------------------------------------------------------------------------------
#  Elastic net graphs(boxplots of CV estimates)
   le_index <- 7
   re_index <- 0
   filename<- "P:\\elastic_dev_05_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   box <- boxplot(t(cv_pred_dev[le_index:(length(lambda_seq)-re_index),]),outline=FALSE,
   main="",col=c(8,8,8,8,2,8,8,8,8,8),  xlab="Log Lambda", ylab="Deviance", names=sprintf("%.2f",
   log(lambda_seq[le_index:(length(lambda_seq)-re_index)])))
   plotCI(x=1:(length(lambda_seq[le_index:(length(lambda_seq)-re_index)])) +0.2,
   y=rowMeans(cv_pred_dev[le_index:(length(lambda_seq)-re_index),]), barcol="blue", lwd=1,sfrac=0.005,
   uiw=sd(t(cv_pred_dev[le_index:(length(lambda_seq)-re_index),])),add=T,col="blue", gap=0,pch=8,cex=1)
   dev.off();

   filename<- "P:\\elastic_cr_05_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   box<-boxplot(t(cv_pred_cr[le_index:(length(lambda_seq)-re_index),]),outline=FALSE,
   main="",col=c(8,8,8,8,8,2,8,8,8,8), xlab="Log Lambda", ylab="Misclassification Rate [%]",
   names=sprintf("%.2f", log(lambda_seq[le_index:(length(lambda_seq)-re_index)])))
   plotCI(x=1:(length(lambda_seq[le_index:(length(lambda_seq)-re_index)])) +0.2,
   y=rowMeans(cv_pred_cr[le_index:(length(lambda_seq)-re_index),]),barcol="blue", lwd=1,sfrac=0.005,
   uiw=sd(t(cv_pred_cr[le_index:(length(lambda_seq)-re_index),])),add=T,col="blue", gap=0,pch=8,cex=1)
   dev.off();
   #coefficients path plot
   filename<- "P:\\elastic_path_05_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=5)
   glm_fit = glmnet(X[train_sample,],y[train_sample], family="binomial",alpha=0.5,standardize=FALSE,nlambda=100)
   plot(glm_fit,xva="lambda", type="l",col="black",pch=19,lwd='0.5',label=F,
   xlim=c(log(glm_fit$lambda[1]),log(glm_fit$lambda[100])))
   mtext("degrees of freedom",side=3,cex=0.9)
   abline(v=-4.11,lty=2,col="red",lwd=2)
   grid(col = "gray", lty = "dotted",nx=NULL,ny=NULL)
   dev.off();


#-------------------------------------------------------------------------------
#Elastic net testing
#-------------------------------------------------------------------------------
   opt_lambda <- exp(-4,81) #optimal value for the regularization parameter
   elastic_output<-elastic_net_test(X[train_sample,],y[train_sample],X[-train_sample,],y[-train_sample],FALSE,1,opt_lambda)

#-------------------------------------------------------------------------------
#Principal component LR training
#-------------------------------------------------------------------------------
   max_pc <- 80 #number of maximal included pc in the model
   runs <- 20
   press_msep <- matrix(nrow = max_pc, ncol = runs)
   press <- matrix(nrow = max_pc , ncol = runs)
```

```
   press_all <- matrix(nrow = max_pc+1 , ncol = runs)

   for (i in 1:runs) {
      pca_output <- princ_comp_train(X[train_sample,],y[train_sample],max_pc,10)
      press[,i]<-pca_output[[2]]
      press_msep[,i]<-pca_output[[1]]
      press_all[,i]<-pca_output[[3]]
   }
   singular<-pca_output[[4]]  #singular values
   press<-rowMeans(press)
   press_all<-rowMeans(press_all)

 #-------------------------------------------------------------------------------
 # R statistic
   press_r<-vector(length=max_pc)
   for (comp in 1:max_pc) {
      press_r[comp]<- press[comp]/press_all[comp]
   }
#-------------------------------------------------------------------------
#  Graphs
   filename<-"P:\\press_pca_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   plot(press, main="",type="b", col="blue",pch=19,ylab="PRESS", xlab="Number of Components")
   dev.off()

   filename<-"P:\\press_r_pca_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   plot(press_r, main="",bg=(as.numeric(press_r > 1)+1),cex=1.0,pch=(as.numeric(press_r > 1)+21),
   type="b",lty=3,ylab="R - Statistic", xlab="Number of Components",col=(as.numeric(press_r > 1)+1))
   abline(h=c(1),lty=2,col="blue",lwd=2)
   text(2,1.05,"Cutoff level", cex=0.8, col="blue")
   dev.off()

   filename<-"P:\\press_msep_pca_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   mean_cv <- rowMeans((press_msep[,1:runs]),na.rm = TRUE) #na.rm removes Na entries
   sd_cv <- sd(t(press_msep),na.rm=TRUE)
   plot(mean_cv, main="",type="b",pch=19,lty=4,cex=1.0,
   ylab="PRESS - MSEP", xlab="Number of Components",ylim=c(min(mean_cv-sd_cv),max(mean_cv+sd_cv)))
   plotCI(x=1:max_pc,y=mean_cv,uiw=sd_cv,add=TRUE,col="black", barcol="blue", lwd=0.5,sfrac=0.005,gap=0.5)
   dev.off()

   pca<-prcomp(X[train_sample,],center=TRUE,scale.=FALSE)
   varexpl<-cumsum(pca$sdev^2)/sum(pca$sdev^2)
   eigenval<-pca$sdev^2   #calculate eigenvalues

   filename<-"P:\\eigenvalues_scen2.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   plot(eigenval[1:60], main="",type="b",lty=3,col=(as.numeric(eigenval[1:60]<mean(eigenval))+1),
   ylab="", xlab="",pch=19,axes=F)
   abline(h=mean(eigenvalues$values),lty=2,col="blue",lwd=2)
   text(6,mean(eigenval)+30,"Avg. Eigenvalue", cex=0.8, col="blue")
   par(new=T)
   plot(varexpl[1:60], main="",type="l",lwd=2,ylab="Cumulative Proportion of Variance Explained",
   xlab="Number of Components")
   grid()
   dev.off()
#-------------------------------------------------------------------------------
#Principal component LR testing
#-------------------------------------------------------------------------------
   opt_pc<-50  #number of optimal pc due to training results
   pca_output_tests <- princ_comp_test(X[train_sample,],y[train_sample],X[-train_sample,],y[-train_sample],opt_pc)

#-------------------------------------------------------------------------------
# General Partial Least Squares training
#-------------------------------------------------------------------------------
   runs<-10
   max_gpls_comp <- 29  #max. number of gpls components
   cv_fold<-10
   cv_pred_dev<-matrix(nrow = max_gpls_comp, ncol = runs)
   cv_pred_cr<-matrix(nrow = max_gpls_comp, ncol = runs)
   for(i in 1:runs) {
      gpls_output<-gen_pls_train(X[train_sample,],y[train_sample],max_gpls_comp,cv_fold)
      cv_pred_dev[1:length(gpls_output[[1]]),i]<-gpls_output[[1]]
      cv_pred_cr[1:length(gpls_output[[2]]),i]<-gpls_output[[2]]
      print(runs)
   }
   gpls_comp_seq<-gpls_output[[3]]
#-------------------------------------------------------------------------------
#  Graphs of training results (boxplots)
   filename<- "P:\\gpls_dev_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   boxplot(t(cv_pred_dev[1:length(gpls_comp_seq),]), main="",col=c(8,2,8,8,8,8,8,8,8,8),outline=FALSE,
   xlab="Number of Components", ylab="Deviance", names=sprintf("%d", gpls_comp_seq[1:length(gpls_comp_seq)]))
```

```
   plotCI(x=1:(length(gpls_comp_seq)) + 0.2,y=rowMeans(cv_pred_dev),uiw=sd(t(cv_pred_dev)),add=T,col="blue",
   barcol="blue", lwd=1,sfrac=0.005,gap=0,pch=8,cex=1)
   dev.off()

   filename<- "P:\\gpls_cr_scen1.eps"
   postscript(filename,horizontal=FALSE,paper="special",width=10,height=7)
   boxplot(t(cv_pred_cr[1:length(gpls_comp_seq),]), main="",col=c(8,2,8,8,8,8,8,8,8,8),outine=FALSE,
   xlab="Number of Components", ylab="Misclassification Rate", names=sprintf("%d", gpls_comp_seq[1:length(gpls_comp_seq)]))
   plotCI(x=1:(length(gpls_comp_seq)) + 0.2,y=rowMeans(cv_pred_cr),uiw=sd(t(cv_pred_cr)),add=T,col="blue",
   barcol="blue", lwd=1,sfrac=0.005,gap=0,pch=8,cex=1)
   dev.off()
#------------------------------------------------------------------------------
# General Partial Least Squares testing
#------------------------------------------------------------------------------
   opt_gpls<-11   #number of optimal components for gpls
   gpls_output_tests <- gen_pls_test(X[train_sample,],y[train_sample],X[-train_sample,],y[-train_sample],opt_gpls)

#------------------------------------------------------------------------------
# Some additional Graphs
#------------------------------------------------------------------------------
#       Age histogram
index<-1:length(dataset$alter1)
age<-alter1[-index[dataset$alter1==0]]
h <- hist(age)
postscript("c:\\test.eps",width=7,height=7,paper="special",horizontal=FALSE)
plot(h,xlab="age",main="")
col<-c("antiquewhite2","antiquewhite3","antiquewhite4","gray20","gray60")

Q5 <- quantile(age, 0.80)
Q5.x <- c(Q5, h$breaks[h$breaks > Q5])
Q5.y <- h$counts[(h$breaks > Q5)[-1]]
rect(Q5.x[-length(Q5.x)], 0, Q5.x[-1], Q5.y, col=col[1], border=NA)
text(Q5+10, max(h$counts), "Q5", srt=90, adj=1.1, cex=0.7)

Q4 <- quantile(age, 0.80)
Q4.x <- c(h$breaks[h$breaks < Q4], Q4)
Q4.y <- h$counts[h$breaks <Q4]
rect(Q4.x[-length(Q4.x)], 0, Q4.x[-1], Q4.y, col=col[2], border=NA)
text(Q4-3, max(h$counts), "Q4", srt=90, adj=1.1, cex=0.7)

Q3 <- quantile(age, 0.60)
Q3.x <- c(h$breaks[h$breaks < Q3], Q3)
Q3.y <- h$counts[h$breaks <Q3]
rect(Q3.x[-length(Q3.x)], 0, Q3.x[-1], Q3.y, col=col[3], border=NA)
text(Q3-3, max(h$counts), "Q3", srt=90, adj=1.1, cex=0.7)

Q2 <- quantile(age, 0.40)
Q2.x <- c(h$breaks[h$breaks < Q2], Q2)
Q2.y <- h$counts[h$breaks <Q2]
rect(Q2.x[-length(Q2.x)], 0, Q2.x[-1], Q2.y, col=col[4], border=NA)
text(Q2-7, max(h$counts), "Q2", srt=90, adj=1.1, cex=0.7)

Q1 <- quantile(age, 0.20)
Q1.x <- c(h$breaks[h$breaks < Q1], Q1)
Q1.y <- h$counts[h$breaks <Q1]
rect(Q1.x[-length(Q1.x)], 0, Q1.x[-1], Q1.y, col=col[5], border=NA)
text(Q1-10, max(h$counts), "Q1", srt=90, adj=1.1, cex=0.7)
dev.off()
#------------------------------------------------------------------------------
#   Test results visualized as barplots
 results_scen1 <- matrix(c(37.28,1.273,0.6851,1.303, 36.50,1.267,0.6909,0,36.02,1.253,0.7003,1.312,
                           35.72,1.250,0.7032,1.287,35.72,1.249,0.7036,1.286), nrow = 5, ncol=4, byrow=TRUE,
                           dimnames = list(c("PCLR","GPLS","El.(0.05)", "El.(0.5)","Lasso"),
                           c("Mis.  Rate [%]", "Deviance", "AUC","AIC")))

 results_scen2 <- matrix(c(38.81,1.290,0.6857,1.439, 37.31,1.266,0.7045, 0, 35.82,1.225,0.6982,1.698,
                           35.82,1.216,0.7161,1.604, 34.33,1.203,0.7196,1.591), nrow = 5, ncol=4, byrow=TRUE,
                           dimnames = list(c("PCLR","GPLS","El.(0.05)", "El.(0.5)","Lasso"),
                           c("Mis.  Rate [%]", "Deviance", "AUC","AIC")))
results<-results_scen1
mybarcol <- "gray20"
par(mfrow = c(2, 2))
 barplot2(results[,1],
         col = c("gray"), width=c(0.1,0.1,0.1,0.1,0.1),space=2,xlab=" ", main = "Misclassification Rate",
         font.main = 4,sub = "", col.sub = mybarcol,cex.names = 0.8, plot.ci = FALSE, ci.l = ci.l,
         ci.u = ci.u,plot.grid = TRUE,ylim=c(min(results[,1]-0.8),max(results[,1])+0.5),xpd=FALSE,ylab="[%]")
 barplot2(results[,2],
         col = c("gray"), width=c(0.1,0.1,0.1,0.1,0.1),space=2, ,xlab=" ",main = "Deviance",
         font.main = 4,sub = "", col.sub = mybarcol, cex.names = 0.8, plot.ci = FALSE, ci.l = ci.l,
         ci.u = ci.u,plot.grid = TRUE,ylim=c(min(results[,2]-0.02),max(results[,2])+0.01),xpd=FALSE)
 barplot2(results[,3],
         col = c("gray"), width=c(0.1,0.1,0.1,0.1,0.1),space=2,horiz=F,xlab=" ",
         main = "AUC", font.main = 4,sub = "", col.sub = mybarcol, cex.names = 0.8, plot.ci = FALSE,
         ci.l = ci.l, ci.u = ci.u,plot.grid = TRUE,ylim=c(0.68-0.001,max(results[,3])+0.005),xpd=FALSE)
```

```
 barplot2(results[,4],
         col = c("gray"), width=c(0.1,0.1,0.1,0.1,0.1),space=2,horiz=F,xlab=" ",
         main = "AIC", font.main = 4,sub = "", col.sub = mybarcol, cex.names = 0.8, plot.ci = FALSE,
          ci.l = ci.l, ci.u = ci.u,plot.grid = TRUE,ylim=c(1.25-0.015,max(results[,4])+0.01),xpd=FALSE)
#*****************************************************************************
```

C.2 function.r

```
dw #*****************************************************************************
# Function File
#-----------------------------------------------------------------------------
#Author: Grosswindhager Stefan
#Last Update: 26.09.09
#Title: Dimension reduction techniques for micro array data
#*****************************************************************************

#-----------------------------------------------------------------------------
# Read the data
#-----------------------------------------------------------------------------
init_data <- function(path,filename) {
           setwd(path)                      #Set the directory
           dataset <- read.csv(filename,header=TRUE, sep = ";",dec=".")   #Load the data from a csv. file
}

#-----------------------------------------------------------------------------
# Elastic net(ridge, lasso,..) training
#-----------------------------------------------------------------------------
elastic_net_train <- function(X,y,n_lambda,std,alpha_val,cv_fold) {

   elastic_fit = glmnet(X,y, family="binomial",alpha=alpha_val,standardize=std,nlambda=n_lambda,dfmax=280)
   cv<-cvsegments(length(y), cv_fold, type = "random")   #get the random cv sets
   cv_pred_dev<-matrix(ncol = length(elastic_fit$lambda), nrow = cv_fold)
   cv_pred_cr<-matrix(ncol = length(elastic_fit$lambda), nrow = cv_fold)

   for (i in 1:length(elastic_fit$lambda))  {
      for   ( k in 1:cv_fold) {
         glm_fit = glmnet(X[-cv[[k]],],y[-cv[[k]]], family="binomial",alpha=alpha_val,standardize=std,
         lambda=elastic_fit$lambda[i])
         y_hat <- predict(glm_fit,newx=X[cv[[k]],],type="response",s=elastic_fit$lambda[i])
         cv_pred_dev[k,i] <- (1/length(y_hat)) * sum( y[cv[[k]]]*log(y_hat) +   (1-y[cv[[k]]])*log(1-y_hat))  #deviance
         y_hat <- sapply(y_hat,classifier)          # 0, 1 loss function   if y_hat 0.5 -> 1 y_hat < 0.5 0
         cv_pred_cr[k,i] <-  (1/length(y_hat))*sum((y[cv[[k]]]*y_hat) + (1-y[cv[[k]]])*(1 - y_hat))
      }
   }
   cv_pred_dev<-colMeans(cv_pred_dev) #calculate column means
   cv_pred_cr<-colMeans(cv_pred_cr)

   return(list(-2*cv_pred_dev,(1-cv_pred_cr),elastic_fit$lambda))  #return the data
}
#-----------------------------------------------------------------------------
# Elastic Net testing
#-----------------------------------------------------------------------------
elastic_net_test <- function(X_train,y_train,X_test,y_test,std,alpha_val,lambda_val) {

   glm_fit = glmnet(X_train,y_train, family="binomial",alpha=alpha_val,standardize=std,lambda=lambda_val)
   y_hat_train <-  predict(glm_fit,newx=X_train,type="response",s=lambda_val) #predict probabilities
   y_hat_test <- predict(glm_fit,newx=X_test,type="response",s=lambda_val)
   degree_f<-glm_fit$df    #get model complexity
   filename<- "D:\\elastic_roc_$alpha_scen2.eps" #filename of figure
   filename<-sub("$alpha",alpha_val,filename,fixed=TRUE)

 return(testing(filename,y_test,y_hat_test,y_train,y_hat_train,degree_f)) #return values
 }

#-----------------------------------------------------------------------------
# Principal component LR training
#-----------------------------------------------------------------------------
princ_comp_train <- function(X,y,max_pc,cv_fold) {

   cv <-cvsegments(length(y), cv_fold, type = "random")
   press_mse<-matrix(nrow = cv_fold, ncol = max_pc)  #initialization
   press<-matrix(nrow = cv_fold, ncol = max_pc)
   press_all<-vector(length=(max_pc+1))
   singular<-vector(length=ncol(X))

   pca<-prcomp(X,center=TRUE,scale.=FALSE)  #perform PCA
   Z<- pca$x[,1:max_pc]    #score matrix
   singular<-pca$sdev      #get singular values

   #calculate 'press all' for r-statistic
   press_all[1]<-sum(X^2) # = PRESS(0)
```

```
   for (comp in 1:max_pc) {
      x_proj <-   (Z[,1:comp,drop=FALSE]) %*% t(pca$rotation[,1:comp])  + outer(rep(1,nrow(X)),pca$center)
      res<-x_proj-X
      press_all[comp+1] <-  sum(res^2)
   }
   for   (k in 1:cv_fold) {
      X_train =  X[-cv[[k]], ,drop=FALSE]
      X_test =  X[cv[[k]], , drop= FALSE]
      Y_train = y[-cv[[k]]]
      Y_test =  y[cv[[k]]]

      pca_train<-prcomp(X_train,center=TRUE,scale.=FALSE)
      Z_train<- pca_train$x[,1:max_pc]    #training score matrix
      Z_test <- (X_test - outer(rep(1,nrow(X_test)),pca_train$center)) %*% (pca_train$rotation[,1:max_pc])
      for (comp in 1:max_pc) {
         x_test_proj <-   (Z_test[,1:comp,drop=FALSE]) %*% t(pca_train$rotation[,1:comp])
         + outer(rep(1,nrow(X_test)),pca_train$center)
         res<-x_test_proj-X_test
         press[k,comp] <-  sum(res^2)

         glm_reg <- glm.fit(cbind(rep(1,nrow(Z_train[,1:comp, drop = FALSE])),
         Z_train[,1:comp, drop = FALSE]),Y_train, family=binomial())
         Y_hat <- exp(glm_reg$coefficients[1] +  Z_test[,1:comp,drop=FALSE] %*% glm_reg$coefficients[-1])/
         (exp(glm_reg$coefficients[1] +  Z_test[,1:comp,drop=FALSE] %*% glm_reg$coefficients[-1])+1)
         res<-Y_hat-Y_test
         press_msep[k,comp] <-  (1/length(res))*sum(res^2)
      }
   }

   press<- colSums(press)
   press<-press/(nrow(X)*ncol(X))
   press_all<-press_all/(nrow(X)*ncol(X))
   press_msep<-colMeans(press_msep)

   return(list(press_msep,press,press_all,singular))
}
#-------------------------------------------------------------------------------
# Principal component LR testing
#-------------------------------------------------------------------------------
princ_comp_test<-function(X_train,y_train,X_test,y_test,num_pc)  {

    pca_train<-prcomp(X_train,center=TRUE,scale.=FALSE)
    Z_train<- pca_train$x[,1:num_pc]
    Z_test <- (X_test - outer(rep(1,nrow(X_test)),pca_train$center)) %*% (pca_train$rotation[,1:num_pc])

    glm_reg <- glm.fit(  cbind(rep(1,nrow(Z_train)),Z_train),y_train, family=binomial())
    y_hat_train <- exp(glm_reg$coefficients[1] +  Z_train %*% glm_reg$coefficients[-1])/
    (exp(glm_reg$coefficients[1] +  Z_train %*% glm_reg$coefficients[-1])+1)
    y_hat_test <- exp(glm_reg$coefficients[1] +  Z_test %*% glm_reg$coefficients[-1])/
    (exp(glm_reg$coefficients[1] +  Z_test %*% glm_reg$coefficients[-1])+1)

    degree_f = num_pc  #degrees of freedom
    filename<- "D:\\roc_$num_pclr_scen1.eps"
    filename<-sub("$num",num_pc,filename,fixed=TRUE)

    return(testing(filename,y_test,y_hat_test,y_train,y_hat_train,degree_f))
}

#-------------------------------------------------------------------------------
# General partial least squares training
#-------------------------------------------------------------------------------
gen_pls_train <- function(X,y,max_gpls_comp,cv_fold) {

   cv<-cvsegments(length(y), cv_fold, type = "random")
   gpls_comp<-seq(1, max_gpls_comp, by=1)
   cv_pred_dev<-matrix(nrow = cv_fold, ncol = length(gpls_comp))
   cv_pred_cr<-matrix(nrow = cv_fold, ncol = length(gpls_comp))
   n_comp=1  #count variable

   for (comp in gpls_comp)  {

      for   (k in 1:cv_fold) {
         X_train =  X[-cv[[k]],,drop=FALSE]
         X_test =  X[cv[[k]], , drop= FALSE]
         Y_train = y[-cv[[k]]]
         Y_test =  y[cv[[k]]]

         gpls_fit <- gpls(X_train, Y_train, K.prov=comp, family="binomial", br=TRUE,link="logit",lmax=50,eps=1e-2)
         Y_hat <- exp(gpls_fit$coefficients[1] + X_test %*% gpls_fit$coefficients[-1])/
         (exp(gpls_fit$coefficients[1] + X_test %*% gpls_fit$coefficients[-1])+1)

         cv_pred_dev[k,n_comp] <- (1/length(Y_hat)) * sum( Y_test*log(Y_hat) +   (1-Y_test)*log(1-Y_hat))  #deviance
         Y_hat <- sapply(Y_hat,classifier)
         cv_pred_cr[k,n_comp] <-  (1/length(Y_hat))*sum((Y_test*Y_hat) + (1-Y_test)*(1 - Y_hat)) #0,1 loss
```

```
      }
      n_comp<-n_comp + 1
   }
   cv_pred_dev=colMeans(cv_pred_dev)
   cv_pred_cr=colMeans(cv_pred_cr)

   return(list(-2*cv_pred_dev,(1-cv_pred_cr),gpls_comp))
 }

#------------------------------------------------------------------------------
# General partial least testing
#------------------------------------------------------------------------------
gen_pls_test<-function(X_train,y_train,X_test,y_test,num_gpls) {

    gpls_fit <- gpls(X_train, y_train, K.prov=num_gpls, family="binomial", br=TRUE,link="logit",lmax=50,eps=1e-2)
    y_hat_train <- exp(gpls_fit$coefficients[1] + X_train %*% gpls_fit$coefficients[-1])/
    (exp(gpls_fit$coefficients[1] + X_train %*% gpls_fit$coefficients[-1])+1)
    y_hat_test <- exp(gpls_fit$coefficients[1] + X_test %*% gpls_fit$coefficients[-1])/
    (exp(gpls_fit$coefficients[1] + X_test %*% gpls_fit$coefficients[-1])+1)

    degree_f = 0 #problems with df of gpls models
    filename<- "P:\\roc_$num_gpls_scen1.eps"
    filename<-sub("$num",num_gpls,filename,fixed=TRUE)

  return(testing(filename,y_test,y_hat_test,y_train,y_hat_train,degree_f)) #return values
}

#------------------------------------------------------------------------------
# Binary classification with cut off level 0.5
#------------------------------------------------
classifier <- function(x) {    # x in [0,1]
   if  (x<=0.5) {
      x = 0
   }
   else
      x = 1
  }
#------------------------------------------------------------------------------
# Sample Stratification
#------------------------------------------------------------------------------
strat <- function(y,trainsize,frac_resp) {
   train_sample <- NULL
   sample_index <- NULL

   while (length(train_sample)<train_size) {
      sample_index <- sample.int(length(y), 1, replace = FALSE)
      while (any(train_sample==sample_index))  {      #without replacement!!
         sample_index <- sample.int(length(y), 1, replace = FALSE)
      }
      if (y[sample_index] ==  1) {
         if (runif(1,0,1)<=frac_resp)  #runif creates a random uniform distributed variable
            train_sample<- c(train_sample,sample_index)
          }
      else  {
         if (runif(1,0,1)<=(1-frac_resp)) {
            train_sample<- c(train_sample,sample_index)
         }
      }
   }
  return(train_sample) #returns the training sample index set
}

#------------------------------------------------------------------------------
# Test block for the methods
#------------------------------------------------------------------------------
testing <- function(filename,y_test,y_hat_test,y_train,y_hat_train,degree_f) {

   #calculate akaike information criterion
   dev <- -2* sum( y_test*log(y_hat_test) +   (1-y_test)*log(1-y_hat_test))
   dev <- dev/length(y_hat_test)
   aic <- dev + (2*degree_f)/length(y_hat_test)

   #ROC curves (calculation and plotting)
   pred_train <- prediction(y_hat_train, y_train)
   pred_test <-  prediction(y_hat_test, y_test)

   perf_train <- performance(pred_train,"tpr", "fpr")
   perf_test <- performance(pred_test,"tpr", "fpr")
   auc <- performance(pred_test,"auc")   #area under roc curve
   acc_roc<- performance(pred_test,"acc")

   postscript(filename,horizontal=FALSE,paper="special",width=7,height=7)
   plot(perf_train,main="",col="red")
   plot(perf_test,col="blue",add=TRUE,print.cutoffs.at=seq(0.1,0.9,0.1),points.pch=19,text.cex=0.8,
```

```
   text.pos=1,text.offset=0.2,points.cex=0.7)
   lines(c(0,1),c(0,1),lty=2,lwd=1)
   legend("bottomright", inset=.05, c("Training Curve", "Test Curve") , lty=c(1,1),col=c("red","blue"), horiz=FALSE)
   grid(lwd=2)
   dev.off()

   y_hat_test <- sapply(y_hat_test,classifier)
   class_rate <- (1/length(y_hat_test))*sum((y_test*y_hat_test) + (1 - y_test)*(1 - y_hat_test))

   return(list(degree_f,aic,auc,(1-class_rate),dev,acc_roc))
}

#------------------------------------------------------------------------------
# Cost Curves
# fp=false positive (yhat=1;y=0); fn=false negative (yhat=0,y=1)
#------------------------------------------------------------------------------
cost_curves <- function(y_test,y_hat_test) {
    #---Explicit Costs---
    pred_test <-  prediction(y_hat_test, y_test)
    perf_cost_unequal <- performance(pred_test,"cost",cost.fp=15,cost.fn=30)
    perf_cost_equal <- performance(pred_test,"cost",cost.fp=18.5,cost.fn=18.5)
    plot(perf_cost_equal,add=F,xlab="Cutoff Level",ylab="Explicit Costs",col="blue",lty=2)
    plot(perf_cost_unequal,add=T,xlab="",ylab="",col="red",lty=1)
    legend("bottomright", inset=.05, c("Equal Costs", "Unequal Costs") , lty=c(2,1),col=c("blue","red"), horiz=FALSE)
    grid(lwd=2)

   #---Expected Costs---
   # perf_fpr_fnr <- performance(pred_test,'fpr','fnr')
   # perf_cost <- performance(pred_test,"ecost")

   # plot(0,0,xlim=c(0,1),ylim=c(0,1),xlab='Probability cost function',
   # ylab="Normalized expected cost",pch="",main="")
   # lines(c(0,1),c(0,1))
   # lines(c(0,1),c(1,0))
   # for (i in 1:length(perf_fpr_fnr@x.values)) {          #plot cost lines
   #    for (j in 1:length(perf_fpr_fnr@x.values[[i]])) {
   #       lines(c(0,1),c(perf_fpr_fnr@y.values[[i]][j], perf_fpr_fnr@x.values[[i]][j]),col="gray",lty=3)
   #    }
   # }
   # plot(perf_cost,add=T,lwd=2,col="blue")
   # plot(perf_cost_random,add=F,xlab="Probability cost function",ylab="Normalized expected cost",col="blue",lty=2)
   # plot(perf_cost_gpls,add=T,xlab="",ylab="",col="red",lty=1)
   # legend("topright", inset=.05, c("Random Classifier", "GPLS Classifier") , lty=c(2,1),col=c("blue","red"), horiz=FALSE)
   # grid(lwd=2)
}
```

Bibliography

[1] Abdi, H.: *Partial least squares regression.* In Lewis-Beck M., Bryman A., F.T. (ed.): *Encyclopedia of Social Sciences Research Methods.* Thousand Oaks (CA): Sage, Nov. 2003.

[2] Aguilera, A.M., M. Escabias, and M.J. Valderrama: *Using principal components for estimating logistic regression with high-dimensional multicollinear data.* Computational Statistics and Data Analysis, 50:1905–1925, 2006.

[3] Akaike, H.: *A new look at the statistical model identification.* IEEE Transactions on Automatic Control, 19(6):716–723, 1974.

[4] Antoniadis, A. and J. Fan: *Regularization of wavelet approximations.* Amer. Statist. Assoc., 96:939–967, 2001.

[5] Baffi, G., E.B. Martin, and A.J. Morris: *Non-linear dynamic projection to latent structures modeling.* Chemometrics and Intelligent Laboratory Systems, 52(1):5–22, 2000.

[6] Bastien, P., V.E. Vinzi, and M. Tenenhaus: *Pls generalised linear regression.* Computational Statistics and Data Analysis, 48(1):17–46, 2005.

[7] Berry, M.J. and G. Linoff: *Data mining techniques : for marketing, sales, and customer relationship management.* Wiley Publishing, 2004.

[8] Bertsekas, D.P.: *Nonlinear Programming.* Athena Scientific, 1995.

[9] Browne, M.W.: *Cross-validation methods.* Journal of Mathematical Psychology, 44:108–132, 2000.

[10] Chatterjee, S., A.S. Hadi, and B. Price: *Regression Analysis by Example.* Wiley Series in Probability and Statistics, 2006.

[11] Cramer, J.S.: *Logit Models from Economics and other Fields.* Cambridge University Press, 2003.

[12] Cramer, R.: *Chemometric Techniques for Quantitative Analysis.* Marcel Dekker, Inc, 1998.

[13] Ding, B.: *Classification using generalized partial least squares*, May 2009. http://cran.r-project.org/web/packages/gpls/index.html.

[14] Ding, B. and R. Gentleman: *Classification using generalized partial least squares.* Journal of Computational and Graphical Statistics, 14(2):280–298, 2005.

[15] Draper, N.R. and H. Smith: *Applied Regression Analysis 3rd edition.* Wiley Publishing, 1998.

[16] Edirisooriya, G.: *Stepwise regression is a problem, not a solution.* In *Proceedings of the Annual Meeting of the Mid-South Educational Research Association*, Nov. 1995.

[17] Efron, B.: *The bootstrap and modern statistics.* Journal of the American Statistical Association, 95(452):1293–1296, 2000.

[18] Efron, B., T. Hastie, and R. Tibshirani: *Least angle regression.* Ann. Statist., 32:407–499, 2004.

[19] Efroymson, R.: *Multiple regression analysis.* In *Mathematical Methods for Digital Computers*, vol. 1, pp. 191–203. Wiley Publishing, 1960.

[20] Fawcett, T.: *An introduction to roc analysis.* Pattern Recognition Letters, 27:861–874, 2006.

[21] Firth, D.: *Bias reduction, the jeffreys prior and glim.* In *Advances in GLIM and Statistical Modelling*, pp. 91–100, New York, Springer.

[22] Fokianos, K.: *Comparing two samples by penalized logistic regression.* Electronic Journal of Statistics, 2:564–580, 2008.

[23] Friedman, J., T. Hastie, H. Höfling, and R. Tibshirani: *Pathwise coordinate optimization.* The Annals of Applied Statistics, 1(2):302–332, 2007.

[24] Friedman, J., T. Hastie, and R. Thibshirani: *The Elements of Statistical Learning 2nd ed.* Springer, 2009.

[25] Friedman, J., T. Hastie, and R. Tibshirani: *Regularization paths for generalized linear models via coordinate descent.* Techn. rep., Department of Statistics, Stanford University, Standford, 2008.

[26] Friedman, J., T. Hastie, and R. Tibshirani: *Lasso and elastic-net regularized generalized linear models*, Jan. 2009. http://cran.r-project.org/web/packages/glmnet/index.html.

[27] Geladi, P. and B.R. Kowalski: *Partial least squares regression: A tutorial.* Analytica Chimica Acta, 185:1–17, 1986.

[28] Genkin, A., D. Madigan, and D.D. Lewis: *Large-scale bayesian logistic regression for text categorization.* Techn. rep., Rutgers University, Piscataway, 2004.

[29] Giombini, G. and J. Szroeter: *Quasi akaike and quasi schwarz criteria for model selection: A surprising consistency result.* Economics Letters, 75:259–266, 2007.

[30] Haisten, M.: *The real-time data warehouse: The next stage in data warehouse evolution*, 1999.

[31] Han, J. and M. Kamber: *Data Mining: Concepts and Techniques 2nd edition.* Morgan Kaufmann, 2006.

[32] Hand, D.J., P. Smyth, and H. Mannila: *Principles of data mining.* MIT Press, 2001.

[33] Havlicek, H.: *Lineare Algebra II.* Institut für Geometrie Technische Universität Wien, 2003.

[34] Helland, S.I.: *On the structure of partial least squares regression.* Communications in Statistics, Simulation and Computation, 17(2):581–607, 1988.

[35] Himes, D.M., R.H. Storer, and C. Georgakis: *Determination of the number of principal components for disturbance detection and isolation.* In *Proceedings of the American Control Conference*, vol. 2, pp. 1279–1283, Baltimore, Maryland, June 1994.

[36] Hoerl, A.E. and R.W. Kennard: *Ridge regression: Applications to nonorthogonal problems.* Technometrics, 12(1):69–82, 1970.

[37] Holte, R.C. and C. Drummond: *Cost-sensitive classifier evaluation.* In *International Conference on Knowledge Discovery and Data Mining*, pp. 3–9. ACM, 2005.

[38] Hosmer, D.W. and S. Lemeshow: *Applied Logistic Regression.* Wiley Publishing, 200.

[39] Höskuldson, A.: *Pls regression methods.* Journal of Chemometrics, 2:211–228, 1988.

[40] Hubert, M. and S. Engelen: *Fast cross-validation of high-breakdown resampling methods for pca.* Computational Statistics and Data Analysis, 51:5013–5024, 2007.

[41] Inmon, W.H.: *Building the Data Warehouse 4th edition.* Wiley Publishing, 2005.

[42] Jolliffe, I.: *Principal Component Analysis, Second Edition.* Springer, 2002.

[43] Kavšek, B.: *Partial least squares regression and its robustification.* Master's thesis, Technische Universität Wien, May 2002.

[44] Kohavi, R.: *A study of cross-validation and bootstrap for accuracy estimation and model selection.* In *International Joint Conference on Artificial Intelligence*, 1995.

[45] Kraemer, N.: *An overview on the shrinkage properties of partial least squares regression.* Computational Statistics, 22:249–273, 2007.

[46] Krzanowski, W.J. and P. Kline: *Cross-validation for choosing the number of important components in principal component analysis.* Multivariate Behavioral Research, 30(2):149–165, 1995.

[47] Larose, D.T.: *Discovering knowledge in data : an introduction to data mining.* Wiley Publishing, 2005.

[48] Larose, D.T.: *Data Mining Methods and Models.* Wiley Publishing, 2006.

[49] Manne, R.: *Analysis of two partial-least-squares algorithms for multivariate calibration.* Chemometrics and Intelligent Laboratory Systems, 2:187–197, 1987.

[50] Marx, B.D.: *Iteratively reweighted partial least squares estimation for generalized linear regression.* Technometrics, 38(4):374–381, 1996.

[51] Marx, B.D. and E.P. Smith: *Principal component estimation for generalized linear regression.* Biometrika, 77(1):23–31, 1990.

[52] Mas-Colell, A., M.D. Whinston, and J.R. Green: *Microeconomic Theory.* Oxford University Press, 1995.

[53] Mertens, B., T. Fearn, and M. Thompson: *The efficient cross-validation of principal components applied to principal component regression.* In *Statistics and Computing*, vol. 5, pp. 227–235. Springer Netherlands, Sept. 1995.

[54] Park, M.Y. and T. Hastie: *L1-regularization path algorithm for generalized linear models.* J. R. Statistical Society B, 69(4):659–677, 2007.

[55] Ponniah, P.: *Data Warehousing Fundamentals - A Comprehensive Guide for IT Professionals.* Wiley Publishing, 2001.

[56] Powers, D.A. and Y. Xie: *Statistical Methods for Categorical Data Analysis.* Academic Press, 1999.

[57] Pregibon, D.: *Logistic regression diagnostic.* The Annals of Statistics, 9(4):705–724, 1981.

[58] Puhr, C.: *The clinical data warehouse.* Master's thesis, Medizin Universität Wien, 2002.

[59] Schwarz, G.: *Estimating the dimension of a model.* The Annals of Statistics, 6(2):461–464, 1978.

[60] Sellin, N.: *Partial least squares modeling in research on educational achievement.* Reflections on Educational Achievement, pp. 256–297, 1995.

[61] Sing, T., O. Sander, N. Beerenwinkel, and T. Lengauer: *Rocr: visualizing classifier performance in r.* Bioinformatics, 21(20):3940–3941, 2005.

[62] Stein, C.M.: *Estimation of the mean of a multivariate normal distribution.* The Annals of Statistics, 9(6):1135–1151, 1981.

[63] Stone, M.: *Cross-validatory choice and assessment of statistical predictions.* Journal of the Royal Statistical Society, 36:111–147, 1974.

[64] Stone, M. and R.J. Brooks: *Continuum regression: Cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal components regression.* Journal of the Royal Statistical Society. Series B (Methodological), 52(2):237–269, 1990.

[65] Tibshirani, R.: *Regression shrinkage and selection via the lasso.* Journal of the Royal Statistical Society. Series B (Methodological), 58(1):267–288, 1996.

[66] Valle, S., W. Li, and S.J. Quin: *Selection of the number of principal components: The variance of the reconstruction error criterion with a comparison to other methods.* Industrial and Engineering Chemistry Research, 38:4389–4401, 1999.

[67] Wall, M.E., A. Rechtsteiner, and L.M. Rocha: *Singular value decomposition and principal component analysis.* In D.P. Berrar, W. Dubitzky, M.G. (ed.): *A Practical Approach to Microarray Data Analysis*, pp. 91–109. Kluwer: Norwell, Mar. 2003.

[68] Wehrens, R. and B.H. Mevik: *The pls package: Principal component and partial least squares regression in r.* Journal of Statistical Software, 18(2):1–24, 2007.

[69] Witten, I.H. and E. Frank: *Data mining : practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[70] Wold, S.: *Cross-validatory estimation of the numbers of components in factor and principal components models.* Technometrics, 20:397–405, 1978.

[71] Wooldridge, J.M.: *Econometric Analysis of Cross Section and Panel Data.* The MIT Press, 2002.

[72] Wu, T.T. and K. Lange: *Coordinate descent algorithms for lasso penalized regression.* The Annals of Applied Statistics, 2(1):224–244, 2008.

[73] Yang, Y.: *Can the strengths of aic and bic be shared?* Techn. rep., Department of Statistics, Iowa State University, 2003.

[74] Zou, H. and T. Hastie: *Regularization and variable selection via the elastic net.* J. R. Statist. Soc. B, 67(2):301–320, 2005.

[75] Zou, H., T. Hastie, and R. Tibshirani: *On the "degrees of freedom" of the lasso.* Technical report, Stanford University, 2001.