Chapter 5
Data Mining for Cross-Sectional Data

Abstract This chapter investigates methods for analyzing cross-sectional data, i.e., data which are represented in matrix form, where each row represents one process instance. The analysis methods can be grouped into supervised learning methods, also known as predictive analysis, and unsupervised learning. Under the term predictive analysis, we summarize analytical techniques for regression and classification, whereas in case of unsupervised learning, we present methods for cluster analysis. Section 5.1 gives an introduction to supervised learning and Sects. 5.2 and 5.3 present a number of techniques for regression and classification. Section 5.4 treats principles of unsupervised learning and techniques for cluster analysis.

5.1 Introduction to Supervised Learning

In this section, we will follow the approach of Chapter 7 in [14] where one can find a more detailed exposition of the subject. Our knowledge for supervised learning is provided by data in the cross-sectional view of process instances. Given this fact, we distinguish two types of variables: a number of *input variables* $X = (X_1, X_2, \ldots, X_p)$ and one *output variable* Y. The variables define the columns of the data matrix, and lowercase letters (y_i, x_i) are used for the rows of the matrix representing observed values of the i-th process instance. Note that x_i is a p-dimensional vector, but we omit double indices. The number of observed instances is denoted by N, and we frequently use the term *cases* or *observations* for the observed process instances. As denotation for an N-dimensional vector of the output values for all cases, we use $\mathbf{Y}$. $\mathbf{X}_r$ stands for the observed values of the r-th input variable X_r and $\mathbf{X}$ for the $N \times p$ matrix of all observed values of the input variables. Scalar multiplication for p-dimensional vectors u and v will be denoted by $u^T v$.

The analytical goals in supervised learning are the goals defined in Sect. 1.2.4 as estimation and classification. The distinction in supervised learning is formally defined by the scale type of the output variable Y.

W. Grossmann, S. Rinderle-Ma, *Fundamentals of Business Intelligence*,
Data-Centric Systems and Applications, DOI 10.1007/978-3-662-46531-8_5

Analytical Goals in Supervised Learning

- *Regression:* In this case, the output variable Y is a quantitative variable interpreted as *response*, and we want to learn a regression function $Y = f(X)$ from the data $(\mathbf{Y}, \mathbf{X})$. The inputs are frequently called *explanatory variables*.
- *Classification:* Here, the output variable Y is a qualitative variable interpreted as class identifier, and we want to learn a classification function $Y = f(X)$ from the data $(\mathbf{Y}, \mathbf{X})$, which allows a group assignment for the inputs.

In the formulation above, we used capital letters for denotation of the generic relationship between input variables and output variables. In case of the application of this relationship to a specific instance, we use small letters, i.e., $y = f(x)$. The term supervised learning refers to the fact that we have data from the outputs which allow us to "learn" the function f from the data $(\mathbf{Y}, \mathbf{X})$. The learned function can be used afterwards to predict the output value for a new case with input values x_{new}. In case of estimation, the term *predictive modeling* is sometimes used for supervised learning. This indicates that one aims at predicting an output for a given input.

Successful learning of the function has to take into account that we only have partial information, because the data usually are a sample of all possible process instances (cf. Sect. 2.4.3). Moreover, as explained in Chap. 1, the cases are mostly generated by customers and show a variability due to incomplete knowledge about customer behavior and motivations for customer decisions. This implies that we have to use statistical or probabilistic structures in the modeling task. The standard approach in modeling is the definition of a class of model structures $\mathscr{F}$ as candidates for the function f. Typical examples of model classes are linear functions in the input variables, decision trees based on the values of input variables, or classes of probability distributions with unknown parameters. In case of estimation, the standard interpretation for the model class is that the model function $f(X)$ represents the expected (average) behavior of customers. In case of classification, the class assignment $Y = f(X)$ is usually not deterministic but defines a conditional probability $p(Y|x)$ for the classes given the input variables x.

Basically, we can understand the analysis task as finding the "best" function f within the class to achieve the analytical goal. The term "best" is made precise by using the concepts of statistical decision theory. This means that we define a loss function $L(Y, f(X))$ which measures the error when we predict the output Y by the function $f(X)$ and define the risk as expected loss $E[L(Y, f(X))]$. A well-known example of a loss function is the squared loss function $L(Y, f(X)) = (Y - f(X))^2$. In this case, the risk is the expected square error. Using the risk as a quantitative measurement for the error, the best model within the class is defined by that function f which minimizes the risk.

Because our data represent only a sample of all possible cases, we cannot calculate the exact risk. We have to use the empirical risk defined by

$$R_{\text{emp}} = \frac{1}{n} \sum L(y_i, f(x_i)). \tag{5.1}$$

The analysis task can be formulated as the following minimization problem: Find a function $\hat{f}$ such that

$$\frac{1}{N}\sum L(y_i, \hat{f}(x_i)) = \min_{f \in \mathscr{F}} \frac{1}{N}\sum L(y_i, f(x_i)). \tag{5.2}$$

The empirical risk is also called *training error*, because it depends on the data from which we learn the function.

In BI applications, we are not so much interested in minimizing the empirical risk but in finding a realistic estimate for the true risk $E[L(Y, \hat{f}(X))]$ of the model that measures the predictive power of the model. This true risk is also called *test error* or *generalization error* of the model. The two terms can be used interchangeably, but we prefer the term test error. The relation between training error and test error depends not only on the data and the definition of the loss function in the minimization problem in (5.2) but also on the choice of the model class defined by the analyst. We can use rather small model classes, defined by few parameters or large model classes characterized by certain properties of the function. Examples of such classes have been defined in case of regression in the introduction to statistical modeling in Sect. 2.4.4.

An important concept for the distinction of different model classes is *model complexity*. Intuitively, model complexity is defined by the degrees of freedom of the model, i.e., the number of independent parameters which have to be estimated. For example, if our model is defined as linear function in the input variables, the complexity can be measured by the number of input variables used for prediction. In the case of decision trees, the complexity can be defined by the number of nodes of the tree.

To understand the influence of model complexity on the test error, let us consider the regression goal and the squared loss function. Due to the fact that we do not know the class in advance, the test error consists of three components: the first component, called the *irreducible error*, is caused by the variability of the data; the second component, called the *variance*, is caused by the fact that our data is only a sample; and the third component, called *squared bias*, is caused by the choice of the model class.

Model Complexity, Underfitting and Overfitting

- *Underfitting:* A model with low complexity can be estimated with high accuracy from the data, if we measure accuracy by the variance of the estimate. On the other hand, we have to expect that the simple model is not adequate for describing the data, i.e., we have a rather large model bias.
- *Overfitting:* A model with high complexity will be estimated with low accuracy from the data if we measure accuracy by the variance of the estimate. On the other hand, we can expect that the complex model is adequate for describing the data, i.e., we have a low model bias.

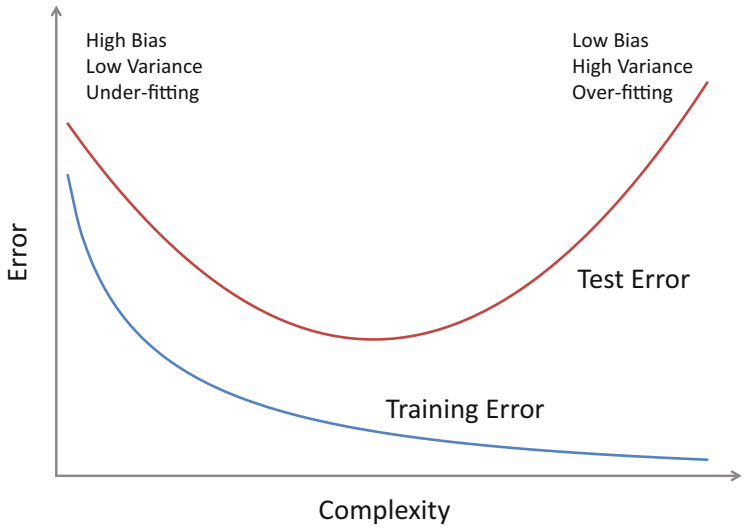

Fig. 5.1 Bias-variance trade-off in model selection

The irreducible error cannot be influenced, but complexity of the model class influences the second and the third component and may cause effects known as *overfitting* and *underfitting*. Balancing overfitting and underfitting in such a way that the test error is minimized is of utmost importance if we want to use the model for prediction of the output variable for new values of the input variables. It is also called *bias-variance trade-off* and schematically shown in Fig. 5.1.

In the best case, the test error can be estimated from theoretical considerations, which give bounds for the generalization error. In the following sections, we will mention results in this direction, which often rely on large sample approximations and are many a time of limited use for practical applications. Hence, one is interested in the empirical evaluation of different possible models with respect to the test error. Looking at this problem from the modeling perspective in Sect. 2.1, we can understand supervised learning as an application of the modeling approach called *models of data*.

If sufficient data are at hand, a general method for model selection and model assessment is splitting the data as follows:

Model Selection and Estimation of Test Error

1. *Data splitting:* Divide the data into a training sample, a validation sample, and a test sample. Frequently, 50 % for the training sample, 25 % for the validation sample, and 25 % for the test sample are recommended.
2. *Model learning:* Use the training data to learn the parameters of models.
3. *Model selection:* Use the validation sample to select an appropriate model.
4. *Model evaluation:* Use the test sample to estimate the test error.

Splitting in three subsamples can be simplified into splitting into two data sets by using techniques which allow the combination of the training and validation step in one single step. These techniques are based either on analytical techniques or on the reuse of the data. In such cases, a splitting rule of 70 % training sample to learn the model and 30 % test sample is frequently recommended. Moreover, in the best case, such techniques allow an estimate of the generalization error. We will explain the different techniques in connection with the models in the following sections.

5.2 Regression Models

After general considerations, we will discuss linear regression models as the most important standard model, introduce the basics of neural networks, and investigate two basic smoothing methods.

5.2.1 Model Formulation and Terminology

Linear regression models are used if we want to predict a metric response variable Y, for example, a KPI, depending on a number of predictor variables $X = (X_1, X_2, \ldots, X_p)$. The model is defined by the equation

$$Y = f(X) + \varepsilon \tag{5.3}$$

where the deterministic component $f(X)$ is the model for the expected behavior of the response Y and ε is a random component describing the variation of the observed data not explained by the model. As model classes for the function f, one can define the classes introduced in Sect. 2.4.4 in connection with the introduction of regression models. The standard assumption for the random component ε is a normal distribution with zero mean. With respect to the class of model functions, the most important class are linear functions in the input variables. This case is treated in Sect. 5.2.2. Alternative analytical techniques allowing more general specifications of model functions are so-called *nonparametric models*. A well-known technique are *neural nets* introduced in Sect. 5.2.3.

Another technique for nonparametric models is *nonparametric regression*. Such methods are of special interest if one wants to visualize the relationship between one input and one output by a smooth function, sometimes called *smoother*. Consequently, software for data visualization offer different smoothers for the visualization of trends in scatter plots (cf. Sect. 4.4.3). We will briefly discuss kernel estimates in Sect. 5.2.4 and smoothing splines in Sect. 5.2.5 as essential techniques, which are frequently used in other models. Generalizations to functions of more than one input variable use the concept of additive models, which define the regression function as an additive model of functions of the individual variables [13]. This

approach avoids the curse of dimensionality. Other approaches to nonparametric regression are functional models. For a comprehensive account, see [22].

The following template contains a summary of the standard procedures for regression models.

Template: Regression Analysis

- **Relevant Business and Data:** Customer behavior represented as cross-sectional data for process instances
- **Analytical Goals:**
  - Prediction of the response function describing the relationship between input variables and output variables
  - Prediction of output values for new input values
- **Modeling Tasks:** Define a model class, for example, linear models or nonparametric models as considered in Sect. 2.4.4
- **Analysis Tasks:**
  - *Splitting data:* Split the data randomly into one set for training and validation and one set for testing the model
  - *Model estimation:* Estimate candidate models by solving the minimization problem for the empirical risk for the training data
  - *Model assessment:* Assess the quality of the model using residual analysis
  - *Model selection:* Select the best model from the candidates using either theoretical considerations or data-oriented methods
- **Evaluation and Reporting Task:** Evaluate the selected model using the test error for the test data

The estimation of the models under consideration is usually defined as a minimization problem for the empirical risk. The estimated response function is denoted by $\hat{f}(X)$ and the predictions for the observations are denoted by $\hat{y}_i = \hat{f}(x_i)$. The deviations $r_i = y_i - \hat{y}_i$ of the observed values from the predictions are called *residuals*. The definition of the empirical risk of the estimate is, in most cases, based on the squared loss function

$$R_{\text{emp}} = \frac{1}{n} \sum L(y_i, \hat{f}(x_i)) = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2. \tag{5.4}$$

However, other specifications are possible, for example, absolute deviations if one is interested in a robust estimation in order to reduce the influence of outliers in the data (cf. [20]).

After the estimation of a model, one has to do model assessment which evaluates the empirical risk and checks the properties of the residuals. The methods for model selection depend on the chosen model and can be based either on theoretical considerations or on the data. We will treat these methods in connection with the models.

For the evaluation of the model, the standard procedure is splitting the data into a training set and a test set. The basic quantities for evaluation are the residuals of the test data is the *mean square error* of the test data. If we use a set of M test data $(y_i^{\text{test}}, x_i^{\text{test}})$, the residuals of the test data are defined by $r_i^{\text{test}} = y_i^{\text{test}} - \hat{y}_i^{\text{test}}$, and the mean square error is defined by

$$\text{MSE}_{\text{test}} = \frac{1}{M} \sum L(y_i^{\text{test}}, \hat{f}(x_i^{\text{test}})). \tag{5.5}$$

5.2.2 Linear Regression

Linear regression is the most prominent predictive model for finding the value of an output variable Y in dependence on the input variables X, and it is well investigated. We sketch only few important aspects for applications. A detailed application-oriented treatment with emphasis on computational aspects is included, for example, in [7].

Modeling Task

Given the input variables, the model class is defined by linear functions in the input variables, i.e.,

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon. \tag{5.6}$$

The interpretation of the models and the parameters is obvious in the case of quantitative input variables. The coefficients measure the effect per unit of the variable on the response. In the case of qualitative inputs (e.g., gender), the coefficients measure the effect of the categories represented by the dummy variable (cf. the description of dummy variables in Sect. 2.4.4). The selection of models within this class can be formulated as selection of relevant input variables, i.e., we understand the input variables as possible candidates for the model and select a subset by defining the coefficients for the variables not present in the model by 0.

Model Estimation

The estimation task is accomplished by the method of least squares; i.e., based on the observations $(\mathbf{Y}, \mathbf{X})$ for the training data, we define the estimate of the parameters in such a way that the empirical risk, defined as squared distance of

the estimated responses $\hat{y}_i$ from the observed responses y_i, is minimized:

$$\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 = \min_{\beta} \sum_{i=1}^{N}(y_i - x_i^T \beta)^2. \tag{5.7}$$

Model Assessment

Model assessment investigates the explanatory power of the model and checks the compliance of the model assumptions.

Assessment of Explanatory Power

The assessment of the explanatory power is based on the decomposition of the sum of squares of the overall deviation from the mean into the explained sum of squares and the residual sum of squares (or "errors"):

$$\sum_{i=1}^{N}(y_i - \bar{y})^2 = \sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2 + \sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{5.8}$$

Here, $\bar{y}$ denotes the overall mean of the data, corresponding to a model with no explanatory variables.

Parallel to this decomposition, we have a decomposition of the *degrees of freedom*, which is interpreted as follows: Altogether, N observations define N degrees of freedom, and the estimation of each parameter decreases the degrees of freedom by 1. Hence, for the squared deviations from the mean, we have altogether $N - 1$ degrees of freedom; for the explained sum of squares based on p variables, p degrees of freedom; and for the residual sum of squares, $(N - p - 1)$ degrees of freedom.

This decomposition defines two standard measures for model assessment: the first one is the *multiple R-squared*, which measures the proportion of the explained sum of squares from the total sum of squares. The second one is the *F-statistic*, which is an overall measure in how far the average explained sum of squares exceeds the average residual sum of squares. If the model assumptions hold, the well-known F-test can be used for testing the null hypothesis that the model gives no significant explanation for the data against the alternative hypothesis that the model gives a significant explanation for the data.

Besides these two overall measures, one can test hypotheses about the effects of input variables. For each parameter component, we test the null hypothesis that the variable has no significant contribution to the explanation of the output (i.e., the parameter has value 0) against the alternative hypothesis that the variable has a significant contribution to the model (i.e., the value of the parameter in the model differs from 0).

Assessment of Model Assumptions

The tests about explanatory power rely essentially on the following model assumptions which guarantee a number of optimal properties of the least squares estimate:

Assumptions for Linear Regression Models

1. *Model specification:* The mean of the function is correctly represented by the model.
2. *Independence:* The observations are independent variables.
3. *Homogeneous variances:* The variances of the error terms are the same for all observations.
4. *Normal distribution:* The error terms are normally distributed.

Hence, an important issue in all practical applications is checking the model assumptions. There exist many methods known under the term regression diagnostics which are based on the residuals $r_i = y_i - \hat{y}_i$.

The basic visual method for model checking is a plot of the residuals against fitted values. Under the assumptions stated above, this plot should show a symmetric scatter around 0. Any kind of visible trend in this plot is an indication that the specification of the model is incorrect. Also the inhomogeneity of variances can be detected, and diagnostics for outliers are possible.

For checking the assumption of the normal distribution of the residuals, a qq plot for the quantiles of the residuals against the quantiles of the normal distribution is useful. If the model assumptions hold, the qq plot should show approximately a straight line.

Another important issue for the quality of the model is multicollinearity which refers to a correlation of the input variables. In the case of models with highly correlated input variables, the variance of the estimates is inflated and results in low prediction power. For visualization tools detecting multicollinearity and further diagnostic methods, we refer to [8].

Model Selection

In Sect. 5.1, we explained that in the case of unknown model specification, balancing between underfitting and overfitting is of utmost importance. A number of strategies are known for the model selection in linear regression. This avoids splitting the data into a training and a validation set. One frequently used method, which is, strictly speaking, not a method of model selection, is variable selection. Variable selection tries to find a subset of explanatory variables which have good explanatory power for the training data. In many practical problems, a complete search for finding the best selection of the input variables is not feasible due to the *curse of dimensionality*. The term curse of dimensionality refers to the computational problems that occur for high-dimensional data. For example, if we have 10 possible explanatory variables, we would have to search the best model out of 2^{10} models. Hence, we have to define specific model selection strategies.

There exist various strategies for variable selection, which, to some extent, also handle the problem of multicollinearity. *Forward selection* is a strategy which builds models stepwise by starting with a model with no explanatory variable, i.e., the model is defined only by the mean. In each selection step, the variable which gives the best additional explanation to the data is added to the model. The process stops if we cannot add any variable with significant additional explanation. *Backward selection* starts with the full model and deletes variables with not significant explanatory power. *Stepwise selection* combines the two approaches by testing after the augmentation of the model with one additional variable, whether anyone of the already existing variables should be removed from the model. In connection with variable selection, it is recommended not to look at the multiple R-squared but on the *adjusted R-squared* as a measure for explanatory power which penalizes the number of parameters in the model.

More theoretically oriented methods for model selection are based on modifying the loss function by a penalization term which balances between fitting and the complexity of the model which is measured by the number of input variables. Two important criteria are the *Akaike information criterion* (AIC) and the *Bayes information criterion* (BIC) defined by

$$\begin{aligned} \text{AIC} &= -2\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \tfrac{d}{N} \\ \text{BIC} &= -2\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + d \cdot \log N. \end{aligned} \tag{5.9}$$

The idea behind AIC is measuring the information loss of a model compared to the "true" model. If we have a number of models at hand, the best choice would be a model which minimizes information loss. In the case of normally distributed errors, an equivalent criterion to AIC is known as Mallows C_p. BIC starts from a prior probability for all models and calculates the posterior probability for the models. We should select the model with the highest posterior probability. A detailed treatment is included in [3].

The above-mentioned strategies for model selection define a model with a definite number of parameters and are available in many statistical software packages as options in regression procedures. Besides these methods, so-called *shrinkage methods* can be used. Shrinkage methods are of special interest in case of a large number of input variables caused by the definition of dummy variables for qualitative input variables (cf. Sect. 2.4.4). In the last years, the *Lasso* became a popular shrinkage method which can be interpreted as a method for continuous variable selection. Instead of minimizing the sum of squares, the Lasso minimizes the penalized loss function

$$\hat{\beta} = \arg\min_{\beta} \sum_{1}^{N}(y_i - x_i^T\beta)^2 + \lambda \sum_{j=1}^{p}|\beta_j|. \tag{5.10}$$

The parameter λ can be interpreted as complexity parameter. In case of $\lambda = 0$, the solution corresponds to the least squares estimate; in case of $\lambda = \infty$, the regression function is estimated by the mean. Another shrinkage method is *ridge regression*, which is probably the oldest shrinkage method. Ridge regression uses the squares of the coefficients for penalization. Details of the Lasso and ridge regression can be found in [14].

We conclude this section with application of regression in context of the CRM use case.

CRM Use Case: Prediction of Sales

Suppose we want to predict the sales of customers from previous sales measured by the sales index `Average_Sales`, duration of the relationship to the enterprise `Duration` and usage intensity of the different services measured by an activity index `Intensity`. As we have already seen in Chap. 4, sales seem to be significantly correlated with the sales index, whereas duration seems to be less important. The results of the model using these variables were nonsatisfying, and we tried to fit a model using the transformed variables $\sqrt{\texttt{Sales}}$, $\sqrt{\texttt{Average\_Sales}}$, `Duration`, and `Intensity`. Using variable selection with AIC dropped `Duration`. The fit of the model is shown in Fig. 5.2. The result is not completely satisfactory, mainly due to the fact that some customers actually show no sales.

Details of the results and the predictive power can be found on the homepage of the book:

www.businessintelligence-fundamentals.com

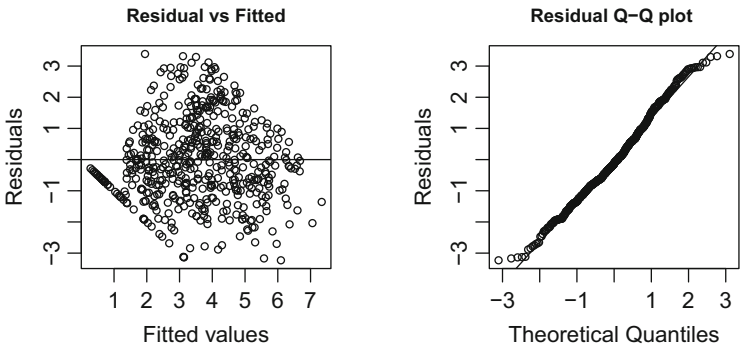

Fig. 5.2 Residual plot and qq plot for sales model (R graphics)

5.2.3 Neural Networks

Neural networks are a modeling technique for the approximation of functions having its origin in the perceptron, one of the first models for learning behavior of the human brain. The model is of interest if one has to find models for problems with limited knowledge about the relation between input and output variables. A comprehensive overview about the application of neural nets for learning may be found in [15].

Modeling Task

Various network topologies and structures have been proposed for different applications, but we consider only the standard backpropagation network with one hidden layer for the approximation of a function $Y = f(X)$. The model is defined by a layered graph with three layers. The first layer is called *input layer* with a number of nodes corresponding to the number of input variables. Usually, a so-called *bias node* is included for modeling the intercept. The middle layer is called *hidden layer* with an arbitrary number of nodes, and the third layer is the *output layer* with a single node representing the response variable. The edges are defined between two adjacent layers bearing weights. Referring to the origin of the model, the nodes of the graph are called *neurons* and the connections between the nodes *synapses*.

CRM Use Case: Prediction of Sales

Figure 5.3 shows a network with three input nodes, two nodes in a hidden layer, and one output layer for modeling the relationship between the output $\sqrt{\texttt{Sales}}$ and the input variables `rootAverage`, `Duration_CR`, `No_activities`, and `No_ServicesUse`. Additionally, the figure shows the bias nodes for modeling the intercepts in the input layer and the hidden layer.

The model works as follows: we take the input values of the variables and propagate the input to the nodes in the hidden layer by using an activation function f applied to the linear combinations of the inputs for the two nodes in the hidden layer:

$$z_i = g_1(w_{0i} + w_{i1}\texttt{rootAverage} + w_{i2}\texttt{Duration\_CR} + w_{i3}\texttt{No\_activities} + w_{i4}\texttt{No\_ServicesUse}) \tag{5.11}$$

Next, the values (z_1, z_2) are used for calculation of the output layer by using an output function g:

$$\sqrt{\texttt{Sales}} = g_2(v_0 + v_1 z_1 + v_2 z_2). \tag{5.12}$$

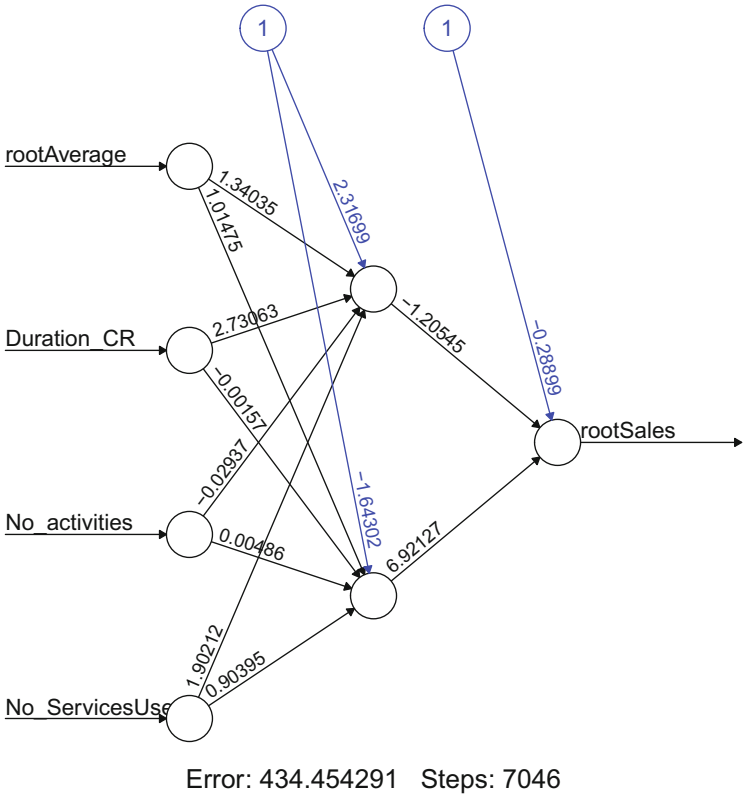

Fig. 5.3 Neural net for prediction of sales (R package `neuralnet`)

For the function g_1, the most popular choice is the *sigmoid function* $f(t) = 1/(1 + \exp(-t))$. For g_2, in case of regression, the identity function is used. If one wants to model an output with values in $[0, 1]$, for example, the prediction of probabilities, the sigmoid function is used.

Model Estimation

For a given number of nodes in the hidden layer, the weights are the parameters which are determined in such a way that the empirical risk of the estimated values for the training data is minimized. With respect to the weights, a local minimum can be found as the zeros of the partial derivatives of the empirical risk by using classical calculus. This problem is solved by the method of the steepest descent, which means that we start from the initial weights and change the weights in the direction given by the negative derivatives. The algorithmic solution uses the following interpretation of the derivatives given by the structure of the problem: The derivatives with respect to the weights v in the second layer can be interpreted as errors caused by the

output layer and the derivatives with respect to weights w in the first layer as errors at the hidden layer depending on the errors at the output layer. This structure allows the realization of the steepest gradient algorithm by the *backpropagation* algorithm:

Algorithm 1: Backpropagation

```
1 Initialization: Start with initial weights, usually defined by a random values;
2 repeat
3     Forward pass: Calculate the predictions for the outputs;
4     Backward pass: Calculate the errors;
      begin
5         Calculate errors at the output layer and derivatives with respect to
          weights v;
6         Calculate errors at the hidden layer using errors at the output layer and
          derivatives with respect to weights w;
7     end
8     Adapt weights in negative direction of the derivatives using a step-size
      parameter;
9 until convergence is reached;
```

Model Assessment

For model assessment, we use the residual sum of squares. For model checking, the basic methods are, as in case of linear regression, a plot of the residuals against the fitted values and a qq plot for checking the distribution of the residuals.

Model Selection and Model Evaluation

From a computational point of view, the configuration of the network is a major task. A well-known result in the theory of neural networks is that one hidden layer is sufficient [16]. For the number of nodes in the hidden layer, the experimentation with different numbers of hidden nodes ranging from 2 up to 100 is proposed. This number depends on the available amount of data. A formal method for avoiding overfitting is *weight decay*, which, similar to ridge regression, uses a penalization of the weight size. For measuring the importance of the different input variables, one can use the concept of *generalized weights*, which measures the importance of an input variable for the estimated output (cf. [10]). One can also use AIC for a comparison between models.

For model evaluation, the standard procedure is splitting the data into a training set and a test set and evaluating the model using the mean square error of the test data.

Let us also mention that in the case of no hidden layer and the identity function as activation function, the neural net corresponds to linear regression. In the case of a sigmoid activation function, a network with no hidden layer corresponds to logistic regression (see Sect. 5.3.2).

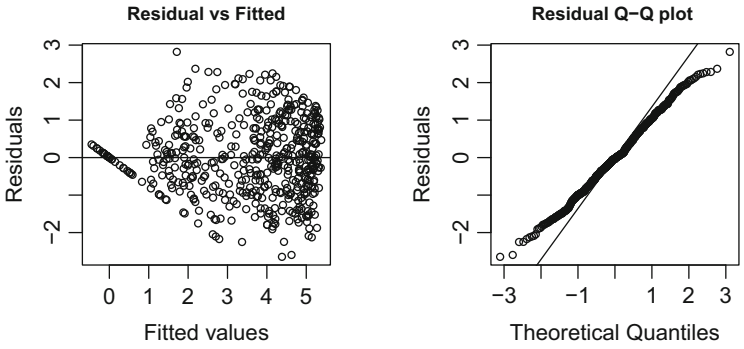

Fig. 5.4 Residuals for neural net (R graphics)

CRM Use Case: Prediction of Sales

We fitted a neural net for the prediction of the sales data. The resulting net is shown in Fig. 5.3. Looking at the residuals in Fig. 5.4, we see that the residuals have a similar structure as in the regression model; however, the normal assumptions for the residuals do not hold.

Details of the calculation can be found on the homepage of the book: www.businessintelligence-fundamentals.com

5.2.4 *Kernel Estimates*

Kernel smoothing defines a model for the relation between the response and one real-valued input variable as a locally weighted mean of neighboring points. The formula of the estimate is given by

$$\hat{f}(x) = \frac{\sum w_j y_j}{\sum w_j} \qquad w_j = \frac{1}{h} K\left(\frac{x - x_j}{h}\right). \tag{5.13}$$

The function $K(x)$ is called *kernel function* which has the properties of a probability density for a distribution with mean 0 and finite variance. One frequently used specification is the normal kernel defined by the normal probability density.

The parameter h is called *bandwidth* which defines the trade-off between smoothness and fit. It can be interpreted as the complexity parameter of the model, which controls the fit to the data by the model. Small values of h will result in a function following close to the data, whereas large values of h will give smooth functions, in extreme cases, the mean.

This model explicitly defines the estimation function and does not use a loss function, but the background of the definition uses a quadratic loss function. For

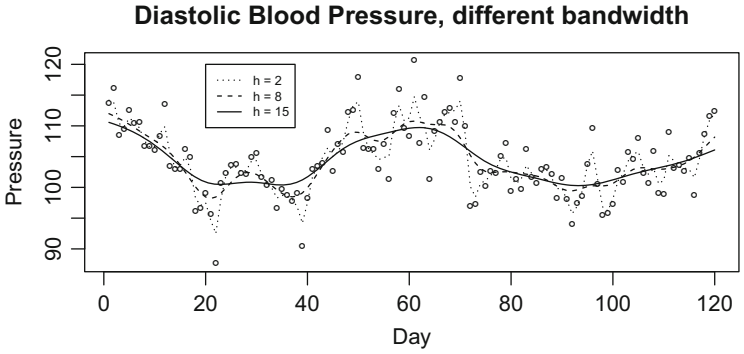

Fig. 5.5 Kernel estimates for blood pressure with different bandwidths (R graphics)

a more comprehensive introduction to the topic, see [1], where one may also find formulas for the calculation of confidence bands for smoothers.

Let us show the application in an example. Note that this example is, strictly speaking, not an example of cross-sectional data but an example of temporal data. This points to the fact that methods for regression can be applied to data of process instances in the state view.

`Preeclampsia Use Case: Prediction of Blood Pressure`

A person's blood pressure often shows high volatility. Thus, a representation of its course over time is of interest. Figure 5.5 shows kernel estimates with different values of bandwidth h for 120 daily measurements of one person's blood pressure. Note that due to the distances between the observations, we have chosen a rather large value for the bandwidth. A bandwidth $h \leq 1$ would almost follow the data points.

Details of the results and the predictive power can be found on the homepage of the book:

www.businessintelligence-fundamentals.com

Model Selection

The determination of an optimal bandwidth in the sense of the predictive power of the model can be obtained by the method of *cross-validation*. Cross-validation calculates the prediction error of an observation y_i which predicts the value by a model $\hat{f}_{(i)}(X)$ using all data points except (x_i, y_i). Obviously, in the case of kernel estimates, this prediction error depends on the choice of the bandwidth h. The optimal parameter h is defined by the minimization of the cross-validation prediction error:

$$\hat{h} = \arg\min_h \sum_{i=1}^{N} (y_i - \hat{f}_{(i)}(x_i))^2. \tag{5.14}$$

It should be mentioned that in some cases, this method for bandwidth determination can produce solutions which are at the boundary of the interval for the bandwidth. In particular, if the data show outliers or if the number of observations is rather small (say less than 100), the method recommends a small bandwidth. This means that the estimate follows close to the data.

5.2.5 *Smoothing Splines*

Modeling Task

Smoothing splines start from the idea that the relation $Y = f(X)$ between two real variables X and Y should be a smooth function. Smoothness of a function f is measured by the second derivative of a function: a function with absolute small second derivative at a peak has a low curvature at the peak, whereas a function with absolute large second derivative has a high curvature at a peak. More precisely, the estimate $\hat{f}$ is defined as the solution of the following minimization problem:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \sum_{i=1}^{N} (y_i - f(x_i))^2 + \lambda \int ((f''(x))^2 dx. \tag{5.15}$$

The interpretation of the equation is that the first term measures the fit of the function to the data, and the second term measures the degree of smoothness. The parameter λ gives the importance of the smoothness and has to be chosen in an appropriate way. If λ is close to 0, we do not care much about smoothness and would approximate the function by a polygon through the data points. If λ is rather large, we approximate the data by a linear regression function, which has second derivative 0. In that sense, we can interpret the choice of λ as a method for controlling model complexity as explained in Sect. 5.1.

Model Estimation

It can be shown that, independent of the value of λ, the solution of the minimization problem is a cubic spline with knots defined by all data points. A cubic spline over the interval $[a, b]$ with K knots, $A = a_1 \leq a_2 \leq \cdots \leq a_K = b$, is defined as a piecewise polynomial of degree 3 in each interval $[a_i, a_{i+1}]$. These polynomials are connected at the nodes in a smooth way which means that the values and the first and the second derivatives of the polynomials on both sides of the knot are identical. Details may be found in [5]. The choice of the parameter λ corresponds to the idea of model selection in regression.

Model Selection

As in the case of kernel estimation, the standard method for choosing λ is cross-validation. For a given λ, we calculate for each observation the prediction error of a

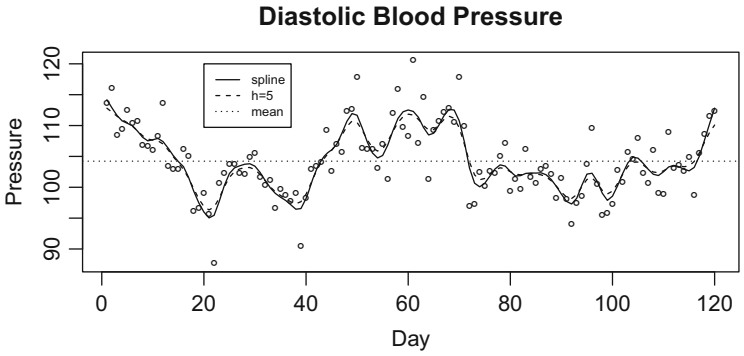

Fig. 5.6 Comparison of smoothing spline and kernel estimate (R graphics)

model $\hat{f}_{(i)}(X)$ estimated from all data points except (x_i, y_i). The optimal parameter λ is defined by the minimization of the cross-validation criterion:

$$\hat{\lambda} = \arg\min_{\lambda} \sum_{i=1}^{N} (y_i - \hat{f}_{(i)}(x_i))^2. \tag{5.16}$$

`Preeclampsia Use Case: Prediction of Blood Pressure`
We applied spline smoothing for the data of diastolic blood pressure and obtained an optimal smoothing parameter $\lambda = 1.3e-6$. Choosing an optimal bandwidth for the kernel estimates did not work. Figure 5.6 shows a smoothing spline for a certain person and a smoothing spline with bandwidth $h = 5$. As one can see, the results are rather comparable. From the graphics, we conclude that the person has a rather constant blood pressure with occasional peaks that are kept in the model. These peaks explain to some extent that the automatic choice of an optimal bandwidth does not work.

Details of the results and the predictive power can be found on the homepage of the book:

www.businessintelligence-fundamentals.com

5.2.6 Summary: Regression Models

Regression models are used for the prediction of the value of a response variable in dependence of a number of explanatory variables. After an assessment of the data by descriptive measures and by visualization, the first step in the analysis is the choice of an appropriate model class. Three different model classes were considered in this section: linear regression models, neural network models, and

nonparametric regression. Using the training data, the possible candidate models are estimated. The criterion for estimation is the minimization of the empirical risk. After estimation, the model assessment has to be done. The most important input for model assessment are the residuals. The selection of a final model out of a number of candidate models is done by different methods. In the case of linear regression, a number of methods are available. From a theoretical point of view, the utilization of information criteria is advisable. In the case of neural networks, the ideas used in linear regression have to be modified and are of more heuristic character. An alternative could be splitting the training data in a training set and a validation set. In the case of nonparametric regression, the method of cross-validation can be used. After the selection of the final model, the predictive power of the model has to be checked by evaluation of the empirical risk for the test data.

5.3 Classification Models

After general considerations on classification methods, this section describes a number of frequently used techniques such as Bayes classifiers, logistic regression, tree-based methods, nearest-neighbor classification, support vector machines, and boosting.

5.3.1 Model Formulation and Terminology

For classification models, the data structure is the same as in regression, but the response variable has only a finite number of values $Y \in \{g_1, g_2, \ldots, g_K\}$ defining the classes. Here, we want to learn a rule how the class membership of an observation can be predicted using the explanatory variables X.

The analytical goal in classification can be formulated in two different ways:

Analytical Goals in Classification

1. *Class assignment:* Estimate the output value for the data by a function of the inputs $Y = f(X)$.
2. *Probability for classes:* Estimate the probability distribution of class membership for an observation given the input variables x: $p_g(x) = P(Y = g|x) = p_g(x)$.

In the case of the second formulation, the assignment to a class is usually done by assigning the observation to the most likely class.

There are different model classes which can be used in classification, which are discussed in the following subsections. We will use probabilistic models, trees, models based on distances, support vector machines, and models based on combination. Probabilistic models use the second formulation of the analytical goals; the other models use the first formulation.

For the determination of the empirical risk, two different loss functions can be used. The first one is the indicator function or $0-1$-loss:

$$L(y, \hat{y}) = \begin{cases} 0, & \text{if } y = \hat{y} \\ 1, & \text{otherwise,} \end{cases} \tag{5.17}$$

and the second one is the *loglikelihood function*, which is also called *cross entropy* or *deviance*:

$$L(y, \hat{f}(X)) = -2 \log(\hat{p}(X)). \tag{5.18}$$

The advantage of $0-1$-loss is that generalizations to different costs for different misclassification can be done easily by replacing 1 by a weight $w_{k\ell}$ which defines the cost if data from true class k are assigned to class ℓ. Such considerations are of interest in many applications, in particular in medicine, but we will not go into details. All software packages for classification have options for allowing such weighting.

The representation of the empirical risk for $0-1$-loss is done by the so-called *confusion matrix* which summarizes the correct and incorrect counts for a certain data set. The rows of the matrix correspond to the actual classes of the data and the columns to the predicted classes:

$$C = \begin{array}{c} \\ \hat{g}_1 \\ \hat{g}_2 \\ \vdots \\ \hat{g}_K \end{array} \begin{array}{c} \begin{array}{cccc} g_1 & g_2 & \dots & g_K \end{array} \\ \begin{pmatrix} n_{11} & n_{12} & \dots & n_{1K} \\ n_{21} & n_{22} & \dots & n_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ n_{K1} & n_{K2} & \dots & n_{KK} \end{pmatrix} \end{array}$$

A correct classification rate corresponds to the percentage of the values in the diagonal of the matrix. In case of more general loss functions, this matrix can be used as basic information about the classification, but in order to find the loss, the values have to be multiplied with the weights for the different misclassification costs.

For the computation of the decision function, one usually starts with a classification problem with two classes. In this case, it is convenient to characterize the classes internally either by the class labels $g = 0$ and $g = 1$ or by $g = -1$ and $g = 1$. Some methods allow immediate generalization from two to more than two

Table 5.1 Terminology in connection with binary classification problems

	Actual class		
Prediction	Positive	Negative	
Positive	**True positive (TP)**	*False positive (FP)*	***Precision = TP/(TP+FP)***
Negative	*False negative (FN)*	**True negative (TN)**	***Negative predicted value =TN/(TN+FN)***
	Sensitivity = TP/(TP+FN)	***Specificity = TN/(FP+TN)***	

Other terms used:
Precision = Positive predictive value
Recall = Sensitivity
False positive rate (false alarm) = 1 − Specificity
False discovery rate = 1 − Precision
Accuracy = (TP + TN)/(TP + TN + FP + FN)

classes, other methods use one of the following strategies for solving problems with more than two classes:

Extending Classification to an Arbitrary Number of Classes

- *One versus the rest:* Perform for each class a classification versus all other classes, and classify the observation to the class with the highest probability in the K classifications.
- *Classification of all pairs:* Perform $\binom{K}{2}$ classifications of all pairs of classes, rank the classes according to the number of assignments, and choose the class with the highest rank.

For model assessment and for the overall evaluation of a classification method, a number of criteria based on the classification matrix are used. Table 5.1 summarizes criteria and terminology in use for the case of two classes. This terminology makes also clear that weighting the different kinds of misclassification is a problem of interpretation of the importance of errors in the domain.

In the case of two classes, the *ROC curve* is another useful tool for assessing the quality of classification. Figure 5.7 shows the typical shape of an ROC curve.

The idea behind the ROC curve is to evaluate a classification method for two classes in dependence of the trade-off between the two types of misclassification. Instead of making the class assignment according to the group with $g = 1$, if $P(1|x) \geq 0.5$, we can use another threshold $t \neq 0.5$. This threshold parameter can be interpreted as a different cost for misclassification for the two classes. If we vary this threshold and draw a curve of the sensitivity against 1-specificity of the classification result, we obtain the ROC curve. The straight line corresponds to the classification without any method. The overall quality of the method can be measured by the area under the curve. In the example depicted in Fig. 5.7, the area under the curve is $a = 0.73$.

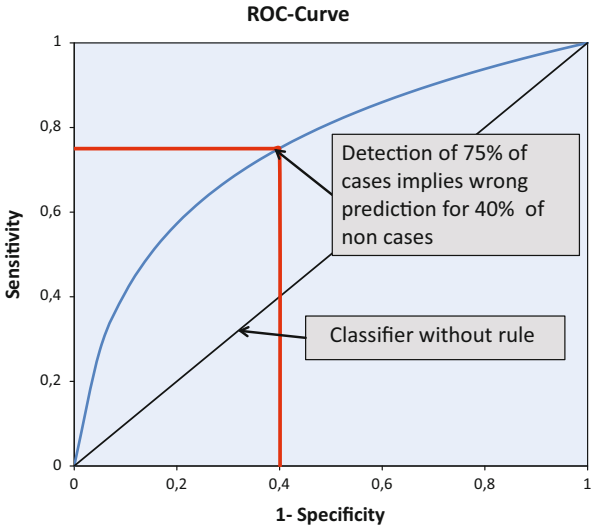

Fig. 5.7 ROC curve for classification of two classes

As already explained in Sect. 5.1, an important issue in classification is model choice in order to avoid overfitting to training data. If sufficient data are available splitting into subsets for training and validation is the best method. In cases of small data sets, a standard method for model selection is k-fold cross-validation, which generalizes the idea of cross-validation introduced in Sect. 5.2.4.

Algorithm 2: k-Fold cross-validation

```
begin
    Divide the training data into k disjoint samples of roughly equal size;
    For each validation sample use the remaining data to construct the
    estimate and estimate the empirical risk for the left out data;
    Compute the prediction error by averaging the empirical risk of the
    validation data;
end
```

A frequently used value for k is $k = 10$. For many classification methods, software packages offer options for integrating cross-validation into the calculations of the classification function.

A summary of the different steps in the analysis of classification problems is included in the following template.

Template: Classification

- **Relevant Business and Data:** Customer behavior represented as cross-sectional data for process instances with a matrix $(\mathbf{Y}, \mathbf{X})$
- **Analytical Goals:**
  - Determination of a function for group membership for the observation
  - Determination of the probability of class membership for the observations given the input values
- **Modeling Tasks:** Definition of a model class for the classification function (cf. Sects. 5.3.2–5.3.6)
- **Analysis Tasks:**
  - *Splitting Data:* Split the data randomly into one set for training and validation, and another set for testing the model
  - *Model Estimation:* Estimate candidate models
  - *Model Assessment:* Assess the quality of the model using the confusion matrix and, if possible, the ROC-curve
  - *Model Selection:* Select the best model from the candidates using either k-fold cross-validation or splitting training data into a training set and a validation set
- **Evaluation and Reporting Task:** Evaluate the selected model using the test error for the test data based on the confusion matrix

In the following sections, we will present different methods. In the description, we will use only the model class, algorithms for finding the classifier, and methods for model selection. Evaluation will be discussed in Sect. 5.3.7.

5.3.2 *Classification Based on Probabilistic Structures*

We will consider two different methods for solving analytically the classification problem: the Bayes approach and logistic regression.

The Bayes Approach

This approach defines a model for the probability distributions in the classes and formulates the classification problem in the context of Bayes decision theory. If we have two classes labeled by $Y = 0$ and $Y = 1$, we assume that the explanatory variables have common distributions in both classes characterized by the densities $p(x|0)$ and $p(x|1)$. Moreover, we know the prior probabilities of the two classes $\pi_0 = P(Y = 0)$ and $\pi_1 = P(Y = 1)$. Given these probabilities, we can define the joint probability of the class and the input variables $p(x, g) = p(x|g)\pi(g)$

and calculate the posterior probability of a class given the input variables using the Bayes theorem (cf. Sect. 2.4.2):

$$P(Y = g|x) = \frac{p(x|g)\pi_g}{p(x)} \qquad g = 0, 1. \tag{5.19}$$

The decision about the class for a new attribute vector x_{new} is given by maximizing the posterior class probability:

$$\hat{y} = \begin{cases} 0, & \text{if } \frac{P(Y=0|x_{\text{new}})}{P(Y=1|x_{\text{new}})} > 1; \\ 1, & \text{if } \frac{P(Y=0|x_{\text{new}})}{P(Y=1|x_{\text{new}})} < 1 \end{cases}. \tag{5.20}$$

It can be shown that this decision is optimal with respect to the risk defined by the misclassification rate. Note that this rule automatically provides the probability of the classes for the vector of explanatory variables. Provision for different costs of misclassification can be included easily by changing the threshold for deciding for group 1 in the decision rule in (5.20). An application for more than two classes is also straightforward by an assignment of an observation to the class with the highest posterior probability.

The crucial point in the application of this rule is the knowledge of the probability densities and the prior probabilities. For the prior probabilities, a standard approach is using the relative frequency of the classes in the training sample. For the probabilities of the explanatory variables, direct estimation from the training data is only feasible if we assume a specific parametric model for the joint densities of all input variables. This approach can be used in the case of normally distributed input variables with the same covariance structure in both groups. It leads to the well-known linear discriminant analysis, which historically stands at the beginning of classification (cf. [14]).

In most practical applications, this approach is not feasible, because the explanatory variables are a mixture of qualitative and quantitative variables. Further, the estimation of the joint probabilities runs into the curse of dimensionality. A standard approach to overcome these problems is the *naive Bayes* method.

Naive Bayes assumes that the input variables are all independent. In this case, the probability distribution is estimated separately for each input variable, and the joint probability is obtained by multiplying all components. The empirical frequencies of the different categories can be used for estimation of qualitative explanatory variables. For quantitative variables, one can use either a parametric model for each variable or density estimation.

Let us show how the approach works for a simple artificial example in the context of a question about customer relationship management (CRM).

Example 5.1 (Customer Behavior in Service Usage) The table in Fig. 5.8 shows the data of 11 customers in the shaded area with respect to the usage of a specific `Service` together with the amount of `Sales`, the `Duration` of customer relationship, and the `Type` of customer (professional or private).

CR-Dur	Sales	User Type	UseService	P(no|x)	P(yes|x)	Decision
10	12	private	yes	0.5004	0.4996	no*
24	36	business	yes	0.0865	0.9135	yes
28	48	business	yes	0.5999	0.4001	no*
45	20	private	yes	0.3121	0.6879	yes
30	34	private	yes	0.0423	0.9577	yes
3	21	private	yes	0,3337	0.6663	yes
1	5	business	no	0.8300	0.1700	no
23	23	business	no	0.5414	0.4586	no
12	49	business	no	0.6672	0.3328	no
35	12	private	no	0.5080	0.4920	no
33	15	private	no	0.4389	0.5611	yes*
12	25	private	??	0.2804	0.7196	yes

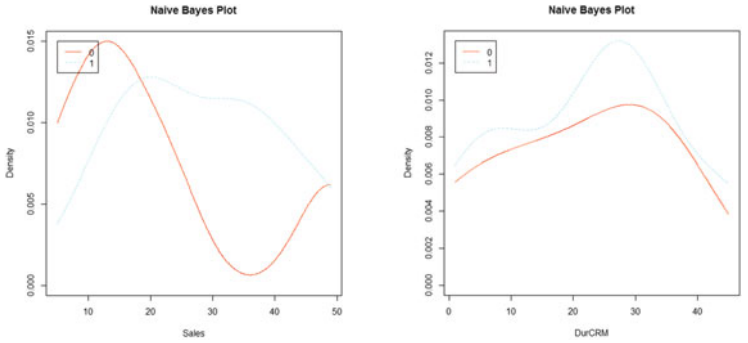

Fig. 5.8 Naive Bayes approach for classification (R graphics)

Our goal is the prediction of the behavior of the customer in the last line. A possible solution is using a nonparametric density estimate for the two quantitative variables and the relative frequency approach for the customer type. Thus, we obtain the posterior probabilities for the classes and the decision about class assignment. As we can see from the training data, eight cases are correctly classified and three are misclassified. The density estimates for `Sales` and `Duration` are shown at the bottom of the figure. Keeping the small number of data in mind, the estimates of the densities are by no means accurate, but the assumption of a normal distribution seems not very realistic in this case.

The naive Bayes approach has been applied successfully in many problems, for example, spam detection which is a simple example of text mining. Note that the approach assumes that the true probability model is within the family of distributions. Model selection is done by the selection of the variables used as predictors for the classes. In general, there is no formal method for model selection as in regression, and we have to use different sets of explanatory variables. Splitting into training and test set is always necessary. A formal analysis of naive Bayes can be found in [11].

Logistic Regression

Logistic regression is a method for modeling the probability of the class of interest $p = P(Y = 1)$ in dependence of a number of explanatory variables. In the case of independent and identically distributed observations of the process instances, the number of instances falling into the class $\{Y = 1\}$ follows a binomial distribution with parameter p. Instead of modeling the dependence of the probability p from the explanatory variables directly, we transform the probabilities into *odds* defined by $\text{odds} = P(Y = 1)/P(Y = 0) = p/(1 - p)$, which are an alternative for defining probabilities in the context of betting (cf. Sect. 2.4.2). Given the odds, one can immediately calculate probabilities by the formula $p = \text{odds}/(1 + \text{odds})$. Next, we transform the odds into the so-called *logits* by $\text{logit} = \ln(\text{odds})$ and define a linear model for the logits by using the explanatory variables:

$$\text{logit} = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p. \tag{5.21}$$

The estimation of the parameters $\beta = (\beta_0, \beta_1, \ldots, \beta_p)$ is done by the method of maximum likelihood, i.e., we choose the parameters in such a way that the likelihood of the data defined by the binomial distribution is maximized. The numerical calculation and assessment of the estimates is accomplished within the framework of generalized linear models. These models transfer the ideas of linear regression to models where a linear model in the explanatory variables is defined for a function of the mean. This function is called *link function*. The overall assessment of the model is based on the deviance as loss function. As in the case of linear regression, a number of diagnostic tools are available for assessing the model. For further details, see [17]. Moreover, techniques for model selection, that are similar to those described for regression in Sect. 5.2.2 can be used.

Predictions for the logits can be transformed into probabilities by using the formula

$$p(X) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_k X_k)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \cdots + \hat{\beta}_k X_p)} \tag{5.22}$$

and the classification rule for new explanatory variables x_{new} is given by

$$\hat{y} = \begin{cases} 1, & \text{if } p(x_{\text{new}}) \geq \text{tr} \\ 0, & \text{if } p(x_{\text{new}}) < \text{tr} \end{cases}. \tag{5.23}$$

The standard choice for the threshold tr is $\text{tr} = 0.5$, but other values can be used in the case of different costs of misclassification for the groups. For an overall assessment of the classification, the ROC curve is used. The generalization to more than two classes is usually achieved by one against the rest, the first method described in Sect. 5.3.1.

Table 5.2 Demonstration example for logistic regression

Duration	ActInd	UserType	Quit	Duration	ActInd	UserType	Quit
5.63	1.93	office(0)	no (0)	6.43	7.6	office(0)	yes(1)
6.39	9.47	office(0)	no (0)	5.55	3.53	private(1)	yes(1)
5.31	9.23	office(0)	no (0)	6.68	3.6	private(1)	yes(1)
5.76	11.67	office(0)	no (0)	3.35	0.23	private(1)	yes(1)
7.12	8.9	office(0)	no (0)	4.31	0.53	private(1)	yes(1)
8.13	9.9	office(0)	no (0)	2.06	2.33	private(1)	yes(1)
4.1	7.27	office(0)	no (0)	3.03	2.5	private(1)	yes(1)
4.29	10.8	office(0)	no (0)	4.78	5.37	private(1)	yes(1)
1.55	4.97	office(0)	no (0)	5.89	1.13	private(1)	yes(1)
0.81	7.2	office(0)	no (0)	4.78	3.83	private(1)	yes(1)
5.25	9.0	private(1)	no (0)	3.83	1.47	private(1)	yes(1)
4.26	8.57	private(1)	no (0)	1.25	2.87	private(1)	yes(1)

A strong point of logistic regression is the interpretation of the parameters β in the model. In the case of a dichotomous input variable, $\exp(\beta)$ measures the change of the odds if the corresponding input variable has the value 1 compared to the odds if the value is 0. This can be interpreted as the increase (or decrease) of the risk for the event of interest if the event defined by $X = 1$ occurs. In the case of qualitative variables defining more than two categories, the interpretation is the change of the risk for a category compared to a reference category. For quantitative input variables, $\exp(\beta)$ measures the change of the odds if the corresponding input variable changes by one unit.

Let us demonstrate logistic regression by a simple example for churn management.

Example 5.2 (Churn Management) Suppose we have 24 observations of customers from which 12 `quit` their relationship to the company in the last year and 12 are still customers. In addition, we know the customers' `UserType` (`private` user or `office` user), activity index `ActInd`, and the `Duration` of the customer relationship. The data are shown in Table 5.2.

Logistic regression for the churn rate showed that `Duration` does not have a significant influence, and the following model was estimated:

$$\text{logit}(\text{Quit}) = 1.385 + 3.058\texttt{UserType} - 0.577\texttt{ActInd} \tag{5.24}$$

The coefficient has the following interpretation: The risk for churning for private users is $\exp(3.058) = 21.3$ times the risk of office users. An increase in activities by one unit reduces the risk of churning by a factor $\exp(-0.577) = 0.56$.

5.3.3 *Methods Using Trees*

The basic idea of tree classification resembles the strategy frequently used in guessing games for terms. By asking a sequence of questions that are answered with yes or no, the candidate tries to limit the number of possible terms until he/she can make a guess with high confidentiality. In combination with this strategy for finding the class membership of the training data, we use a binary tree for modeling purposes. All cases of the training data belong to the root node. In each node of the tree, the data belonging to that node is split into subsets according to the values of one input variable.

Example 5.3 (Customer Behavior in Service Usage) Figure 5.9 shows such a partition for the training data given in Fig. 5.8 for the demonstration of the naive Bayes method.

The root node represents all 11 cases. Using the variable `Sales`, the data are split into cases $1, 7, 10, 11$ with `Sales` < 18 going to the left node and the other cases going to the right node. The data at the left node are split further according to the values of `DurationCRM`. Cases 10 and 11 have a value `DurationCRM` > 22 and go to the right, cases 1 and 7 to the left. At the right node, no further split is necessary, because both cases do not use the service and the terminal node is labeled with 0 indicating `UseService = no`. At the left node, a further split is done according to the value of the variable `DurationCRM` for separating cases 1 and 7. In a similar way, the other cases are partitioned at the right node, and we obtain a perfect classification for the training data.

The classification of the new case in the table in Fig. 5.8 is retrieved according to the decision tree. Because `Sales` $= 25 > 18$, we first go to the right; afterwards, `DurationCRM` $= 12$ indicates to go the left and subsequently to go to the right. Hence, we decide that the customer wants to use the service.

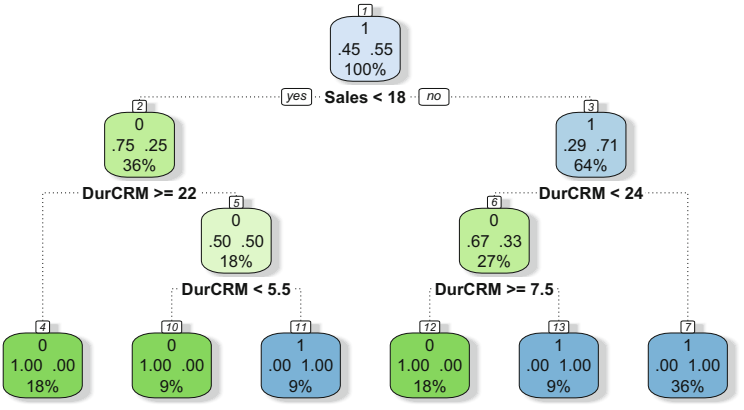

Fig. 5.9 Classification tree for simple example (R package `rattle`)

In this simple example, one can obtain a decision tree probably by experimenting. Defining such a model for realistic training data requires the formal definition of two strategies:

Strategies for the Definition of a Tree Model

1. *Splitting rules:* A strategy for growing the tree defining in each node which variable should be used for splitting together with the threshold for the split.
2. *Pruning rules:* A strategy for pruning the tree in order to avoid overfitting to the training data.

Note that the strategy for defining the model is also the algorithm for defining the classification rule. The most frequently used algorithm for both strategies is *CART* which is the acronym for *classification and regression trees*. The name indicates that the model can also be applied to regression problems. The first systematic treatment of the method was given in [2], and we will briefly discuss the ideas behind splitting and pruning.

Splitting is based on measures for the *impurity* of a node. A node has impurity 0 if only input data of one class belong to the node. If $\hat{p}(g|t)$ denotes the relative frequency of class g at node t, the impurity $Q(t)$ of the node is defined by one of the following two measures:

$$Q(t) = \begin{cases} \sum\sum \hat{p}(i|t)\hat{p}(j|t) = 1 - \sum(\hat{p}(j|t)^2) & \text{Gini Index} \\ Q(t) = -\sum \hat{p}(j|t)\ln(\hat{p}(j|t)) & \text{Entropy} \end{cases}. \tag{5.25}$$

For the split, the variable is used, which minimizes the impurity in the child nodes. Finding this variable can be done by greedy search. In the case of a metric variable, the split is defined by a decision rule in the form ($X < \text{tr}$ OR $X \geq \text{tr}$); in the case of nominal variables, we decide according to the rule ($X = a_t$ OR $X \neq a_t$). The process continues as long as the child nodes contain different classes or the number of cases in the node is too small. In the example tree in Fig. 5.9, the entropy measure was used.

After growing, the tree is pruned for obtaining a simpler decision rule which avoids overfitting. The basic strategy for pruning is to remove splits which lead only to a small improvement of the empirical risk. CART uses a more complex strategy, which allows the pruning of sub-trees by using a penalized risk instead of the empirical risk:

$$R_{\text{pen}}(\alpha) = R_{\text{emp}} + \alpha|T|. \tag{5.26}$$

Here, $|T|$ denotes the number of nodes in the tree measuring the complexity of the model and α is a penalization parameter for complexity. $\alpha = 0$ corresponds to the most complex tree after complete growth, and $\alpha = \infty$ corresponds to a simple model with only a root node. It can be shown that there is a finite number of penalization parameters $\alpha_0 = 0, \alpha_1, \ldots, \alpha_M$ such that for all $\alpha \in (\alpha_i, \alpha_{i+1}]$,

there exists one unique tree with a minimum number of nodes which minimizes the penalized risk $R_{\text{pen}}(\alpha)$. This fact can be used for the efficient computation of a number of candidate tree models with minimum number of nodes from the training data. For selection of the final model, a test set is used or k-fold cross-validation is applied. Details may be found in [23]. The model corresponding to the penalization parameter which minimizes the penalized risk is chosen as the final model.

Some additional features make CART an attractive procedure for applications.

- CART has a mechanism for internal re-weighting of data in case of unbalanced classes, which occurs quite frequently in BI applications.
- CART has a mechanism for handling missing values. Obviously, the tree can be applied only if the variables used in the splitting decisions are known. In case of missing values, CART offers so-called surrogate splits. This means that in tree building, for each knot of the tree, a degree of missingness is computed and alternative variables for the split decision are defined.
- CART automatically adapts to the number of classes and does not require a special mechanism.
- The estimation of the classification probabilities for the training data is rather simple, because we have to calculate only the frequency of observations of the different classes in the terminal nodes.

As weak points of tree classifiers, it is sometimes mentioned that tree building is rather sensitive to the ordering of the input variables and unstable with respect to changes in the training data. In order to overcome such problems, one can perform many classifiers and average the results of these classifiers, for example, by using a majority vote. Usually, only one training data set is available, and additional samples have to be generated. This technique is known as *bootstrap aggregation* or *bagging*. Bagging applies the general method of *bootstrap* to classification problems. The interested reader is referred to [6]. The basic steps in the bagging algorithm are as follows:

Algorithm 3: Bagging algorithm

begin

Given the training data $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, generate B bootstrap samples of size N

$$(\mathbf{x}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{x}^{(2)}, \mathbf{y}^{(2)}), \ldots, (\mathbf{x}^{(B)}, \mathbf{y}^{(B)})$$

from the data by sampling with replacement;

For each bootstrap sample $(\mathbf{x}^{(b)}, \mathbf{y}^{(b)})$ learn a classification tree T_b;

Define the final classification by taking the majority vote of the classifiers;

end

Random forests modify this procedure in such a way that in each split of a bootstrap classification only a random subset of the input variables is used. Such a random selection helps to overcome the problem that the different tree classifiers

may be correlated due to the fact that important predictors are used in all bootstrap classifications.

Sometimes it is mentioned as a disadvantage of the method that the interpretation of complicated trees may be tricky and that trees cannot cope with linear combinations of the variables. With respect to predictive power, other methods, such as neural nets, may be superior. However, in general, CART is considered as one of the most useful methods for classification.

Besides CART, other methods for tree classification exist. The oldest method is CHAID (chi-square automatic interaction detection) developed in the 1960s for partitioning data according to the values of nominal variables. This method does not require pruning. More popular are C4.5 or C5 which also allow growing trees with more than two child nodes. The ideas behind these methods, together with their implementation in Python, may be found in [21].

5.3.4 K-Nearest-Neighbor Classification

This classification method uses a very simple rule for assigning class membership. Given an observation x_{new}, the k closest points to the new observation are determined, and the class assignment is done according to the most frequent class in the k-nearest neighbors. Using instead of the absolute frequencies of the classes in the k-nearest neighbors the relative frequencies allows another interpretation of the classification rule: the classifier is based on estimates of the posterior probabilities of the classes. This shows that nearest-neighbor classifiers have a close connection to Bayes classifiers.

The idea of nearest-neighbor classification is intuitively appealing but depends on two decisions: first we have to define how to measure the distance between the observations and second we have to determine the value of k. The decision about the distances has to take into account the values of the variables. Sometimes it may be advantageous to standardize the variables. Another problem is measurement of the distance between qualitative variables. We will discuss this problem in more detail in Sect. 5.4.

The determination of k has to be done under consideration of the bias-variance trade-off discussed in Sect. 5.1. A small value of k makes the estimate rather unstable, and we can expect a large variance. A large value of k reduces the variance, but the estimate of the posterior probability will be closer to the priors of the classes. Note that if we take k equal to the number of observations of the training set, the estimate will be equal to the sizes of the classes in the training set. This may cause an additional bias for the decision. As it is shown in [14], the decision about k depends on the classification problem and may have substantial influence on the results. Cross-validation is proposed for finding the right k.

The nice property of nearest-neighbor classifiers is that they do not need an explicit training phase. This fact gives the method the name *lazy* classifier. The price for this laziness is the computational effort in finding the k-nearest neighbors. All

the data points of the training set must be kept in the memory and searched for the k-nearest neighbors. This fact justifies the name *memory-based* classifier.

k-nearest-neighbor classifiers have been successfully applied for problems with many different prototypes, for example, in connection with classification problems of time series. We will show an application of that type in Chap. 6.

5.3.5 Support Vector Machines

To understand the ideas behind support vector machines for two classes, it is convenient to label the classes by $Y \in \{-1, 1\}$. For the separation of the classes, we use a linear function in the input variables $f(x) = \sum w_k^T x_k$. The classification rule is then defined by the sign of the function $f(x)$. The weights are defined in such a way that the following two properties hold:

- Explain the training data well
- Achieve maximum separation for correctly classified data

The first property corresponds to the idea of minimizing the training error if we measure explanatory power by the misclassification rate. The second property can be interpreted as generalization error: if we maximize the separation of the classes in the training data, we can expect that new data with a similar behavior will also be classified correctly. In [4, p. 409], an additional explanation of this idea in terms of Popper's principle of falsification is given.

Let us demonstrate this idea by a simple example shown in Fig. 5.10 for two-dimensional input data. The figure shows two possible solutions for a separating line between the two groups. For the evaluation of the two classifications, the concept of *margin* is used. The margin is defined by the points that are closest to the line in the two groups. In the figure, the margin corresponds to the distance between the two

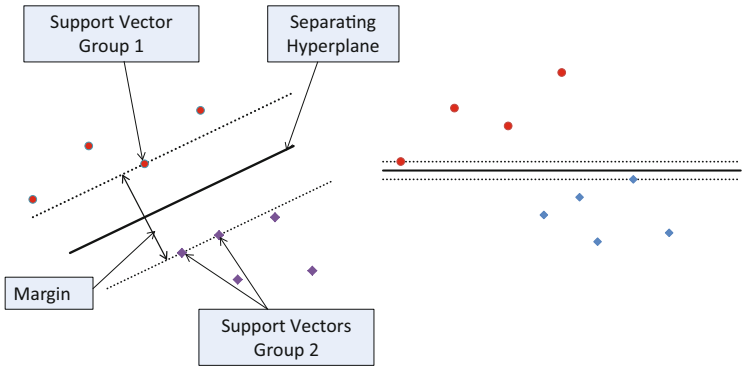

Fig. 5.10 Separation of two classes by hyperplanes. *Left*, a separating line with large margin; *right*, with small margin

parallel dotted lines. Points lying on the dotted lines are called *support vectors*. In the left panel in Fig. 5.10, we have one support vector for group 1 and two support vectors for group 2. In the right panel, there is only one support vector for each group. Obviously, the margin essentially depends on the slope of the separating line defined by the weights and the intercept. The best separating line is defined by that one which maximizes the margin for the training data.

Using standard linear algebra, the task of finding a linear function defined by the weight vector w and the intercept b, which maximizes the margin, is transformed into the following minimization problem:

$$\begin{aligned} &\min \frac{1}{2}||w|| \\ &y_i w^T x + b \geq 1 \qquad i = 1, 2, \ldots n \end{aligned} \tag{5.27}$$

Support vectors are all those points that fulfill the constraint with equality.

The solution of the problem is found by transferring this quadratic optimization problem into its dual problem, which is also a quadratic optimization problem. Solving the problem is numerically demanding, because the number of variables corresponds to the number of training data points. A standard method for obtaining the solution is the *sequential minimization method*, which solves the problem iteratively considering only two multipliers in each iteration.

Using the solution (w^*, b^*) gives the classification rule

$$\hat{y} = \text{sign}(w^T x + b^*). \tag{5.28}$$

Up to now, we made the assumption that for training data, the groups can be separated by a linear function in the space of input variables. The answer whether such a separation is possible leads to the definition of the *Vapnik–Chervonenkis dimension* (VC dimension) for a class of functions, which is defined as follows:

Definition 5.1 (VC Dimension) Given a class of functions $f(x, \theta)$, the VC dimension is defined by the maximum number of points in any configuration that can be shattered by a function in this class.

The interpretation of the term "shattered" is as follows: if we choose points in an arbitrary position and assign the group labels 0 and 1 randomly to the points, then the groups can be separated perfectly by some function of the class. It can be easily seen that for linear functions in the plane, the VC dimension is 3. In case of four or more points, this is not always possible. Generally, the VC dimension of linear functions in p variables is $p + 1$.

This fact shows that using the concept of margin for separation, with linear functions for classification, needs some modifications. Two strategies can be applied. The first one are support vector machines with soft margin as demonstrated in Fig. 5.11. If we use a solid line for separation with margins defined by the dotted lines, then two points are on the wrong side of the separating line. The

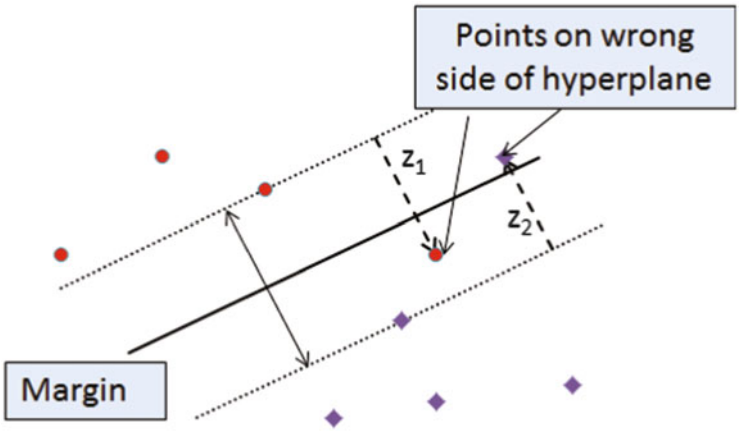

Fig. 5.11 Support vector machine with soft margin

size of violation of separation is given by $z_1 + z_2$. The idea is now to modify the minimization problem in Eq. (5.27) in such a way that misclassification is allowed, but the violation of the constraints penalizes the minimization function. This can be done by introducing new variables ξ_i that measure the misclassification and define the following minimization problem:

$$\min \frac{1}{2}||\mathbf{w}|| + C \sum \xi_i$$
$$y_i \mathbf{w}^t \mathbf{x} + b \geq 1 - \xi_i \qquad \xi_i \geq 0,\ i = 1, 2, \ldots n \tag{5.29}$$

C is a parameter measuring the trade-off between the complexity of the machine and separability.

The second strategy allows nonlinear boundaries in the space of input variables by transforming the problem into a higher dimension. Figure 5.12 shows how this method works for a so-called XOR problem. The left panel shows data in two classes generated by normal distribution with different means. Obviously, a separation by a linear function is not possible. The right panel transforms the data into a three-dimensional space which allows separation in the three-dimensional space by a plane parallel to the (x, y) plane.

For the calculation of this transformation one uses the so-called *kernel trick*. This method changes the definition of the inner product by transforming the data with an inner product kernel. Frequently used kernel functions are *radial basis kernels* defined by

$$H(x, x') = \exp\left\{\frac{-||x - x'||^2}{\sigma^2}\right\}. \tag{5.30}$$

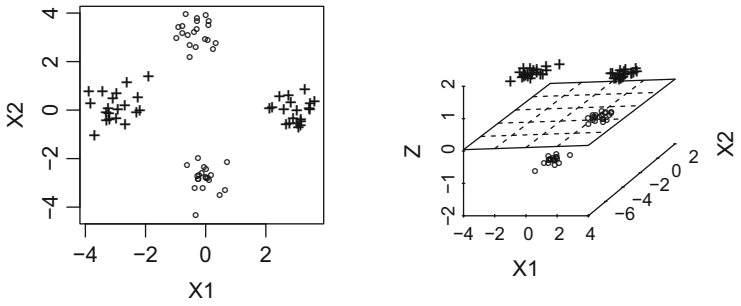

Fig. 5.12 XOR problem and transformation for linear separation (R package `scatterplot3d`)

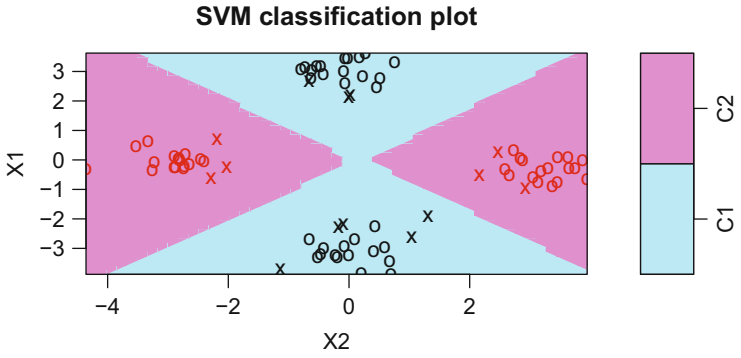

Fig. 5.13 Solutions of the XOR problem with radial basis kernel (R package `e1071`)

The centers of these functions x' and the number of functions correspond to the support vectors. The scaling factor σ is the same for all functions and depends on the data dimension. Other choices of kernels are polynomial kernels. The advantage of the kernel method is that it allows the calculation of the inner products and the distances between observations without explicit transformation of the data. For theoretical details, we refer to [24].

Figure 5.13 shows the solution for the XOR problem with radial basis kernel. The classification uses 14 support vectors, marked by $\times$, and gives a perfect solution. It should be mentioned that the solution depends on the used kernel functions. In this example, the solution with polynomial kernels is much worse.

Using support vector machines for classification allows the determination of the generalization error according to the VC theory. Results may be found in [4]. For model selection, one can use the technique of cross-validation.

The support vector machine is defined for two classes. A standard method for more than two classes is the method one against one. This means that we solve the classification for all pairs and assign that class which has the majority vote.

The calculation of the classification probabilities has to be done separately. One way to do this is by using a logistic distribution which fits best the function defining the decision boundary.

5.3.6 Combination Methods

Combination methods are motivated from the observation that there exists no single best method for all classification problems. One method to overcome this problem is to use random forests as described briefly in Sect. 5.3.3. Another frequently applied method is *boosting*. The theoretical background of boosting is the question on whether it is possible to boost a weak classifier to a strong classifier by repeated application. Under the term weak classifier, we understand a method with classification error only slightly larger than 0.5, i.e., the random guess of the classes taking into account the size of the classes. In terms of the ROC curve for two classes, this means that the area under the curve is only slightly larger than 0.5. Boosting shows that the question has a positive answer using the following basic algorithm:

Outline of the Boosting Algorithm

1. Start with a classifier $f(x)$ for the training data using equal weights w for all observations.
2. Compute modified weights w^* for the data in such a way that misclassified data get a higher weight and correctly classified data get a lower weight.
3. Compute a new classifier $f^*(x)$ for the data with the new weights w^* with the same method.
4 Repeat steps 2 and 3 T times, and define a final classification function as a combination of the T classifiers.

As in the case of support vector machines, the starting point is a classification problem for two classes with training data $((y_1\mathbf{x}_1), (y_2, \mathbf{x}_2) \dots (y_N, \mathbf{x}_N))$ and class identifiers $Y = \{-1, 1\}$. The classification error for a classifier F is defined by the misclassification rate $\varepsilon = \text{card}\{y_i \neq F(\mathbf{x}_i\}/N$. The rationale behind the update for the weights and the weighting of the individual classifiers is based on the *exponential loss* of a classification method which is defined by

$$L_{\exp} = \exp(-yF(x)). \tag{5.31}$$

To find the optimal weights for the individual classifiers in the combination, we consider the loss of an observation if we use a linear combination $F + \alpha f$ of the already existing classifier F and the new classifier f. The parameter α is chosen in such a way that the exponential loss is minimized, and the new weights of the observations are calculated by a Taylor expansion for the exponential loss. A detailed derivation may be found in [25] where one may also find theoretical results

about the generalization error and references to the connection between boosting and logistic regression.

Usually, tree classification with few nodes is used as a weak classifier. This choice allows the usage of cross-validation for the test phase. The generalization for more than two classes is achieved by the method one against the rest.

5.3.7 Application of Classification Methods

In this section, we will apply the above introduced methods to data from the CRM use case (cf. Sect. 1.4.4).

CRM Use Case: Classification of Service Usage

For the comparison of the different methods, we applied the algorithms to predict the usage of a specific service in the CRM use case. As predictors, we used the variables `Sales`, the sales index `SalInd`, the duration of customer relationship `CR-Dur`, the indicators `Student` and `Office` for student users and office users, and the personal characteristics `Age_group` and `Sex`. There were missing values in the variables `Age_gr` (5 %), `Sex` (1 %), `Student` (7 %), and `Office` (7 %). Only customers with a relationship to the company longer than 12 months were considered. The data set encompassed 1,311 customers, from which 748 (75 %) used the service and 563 (43 %) did not use the service. We randomly selected a training data set of 905 customers (approximately 70 %). In the training data set 532 (59 %) used the service and 373 (41 %) made no use of the service. For classification, we used the R procedures for naive Bayes, logistic regression, tree classification, support vector machines, and AdaBoost with the following specifications:

- *Naive Bayes*: All variables were used, and for the quantitative variables, nonparametric density estimation was used because it seemed more realistic than a normal distribution (cf. the results about data visualization in Chap. 4). With respect to missing values, we used the standard strategy, i.e., the omission of cases with missing values.
- *Logistic regression:* For the training data, a model with the variables `Sales`, `Age_gr`, and `CR-Dur` was selected. This selection was supported by the significance of the coefficients as well as by the AIC criterion.
- *CART:* A model for the training data was learned using tenfold cross-validation. It turned out that the splits used the variable `Sales`, the sales indicator `SalesInd`, and the indicator for office.
- *Support vector machines:* We used a specification of a support vector machine with soft margin and a radial basis kernel. For the evaluation of the learned machine from the training data, tenfold cross-validation was used.
- *Boosting:* We trained a model with a maximum number of 10 training rounds using a version of AdaBoost with cross-validation for model selection.

Method	Accuracy	Sensitivity	Specificity	Area under ROC
NaiveBayes, Train	0.782	0.703	0.743	0.817
NaiveBayes, Test	0.665	0.662	0.668	
Logistic, Train	0.714	0.714	0.713	0.790
Logistic, Test	0.685	0.690	0.679	
CART, Train	0.762	0.832	0.662	0.781
CART, Test	0.729	0.824	0.621	
SVM, Train	0.707	0.705	0.710	0.791
SVM, Test	0.672	0.685	0.659	
AdaBoost, Train	0.874	0.705	0.804	0.895
AdaBoost, Test	0.782	0.595	0.695	

Fig. 5.14 Summary of results using different classification methods

The results of the different classifications are shown in Fig. 5.14. For each method, we report sensitivity, specificity, overall rate of correct classification, area under the ROC curve for the training data, and classification results for the test data.

Details of the results can be found on the homepage of the book:
www.businessintelligence-fundamentals.com

5.3.8 Summary: Classification Models

Classification aims at learning a decision rule for the class membership of a new observation from training data. This rule can be formulated either as an estimate of the class indicator or as an estimate of the class probabilities. For the decision of the class membership, a threshold for these probabilities is used. For the evaluation of a decision rule, two methods are the confusion matrix and the cross entropy or deviance. In the case of two classes the ROC curve is a useful tool for assessing the quality of the decision rule.

There are numerous methods for learning the decision rule which are based on different principles. We discussed probabilistic models like naive Bayes or logistic regression, tree-based methods, distance-based methods like nearest-neighbor classifiers, support vector machines which minimize the empirical risk, and combination methods which start with a weak classifier and iteratively improve the decision rule.

Rather independent from the used method, the analysis steps have to follow the general template for supervised learning. We start with descriptive methods for data understanding, split the data in a training and test set, learn a rule from the training set, and assess the decision rule with the test set. The model selection in the training phase depends on the classification method. There are methods like CART or logistic regression that allow integration of model selection in the development of the model. For other methods, the validation of the model has to be done by k-fold

cross-validation or by splitting the training data into a training set and a validation set.

In applications, different methods are applied and tuned to the problem. Afterwards, the most eligible method is used in the deployment phase of the project. In practice, the decision about the classification method depends on a number of factors. In the case of data with missing values, CART may be an interesting option because it offers a mechanism for handling missing values. If one is interested in the interpretation of the rule in subject matter terms, logistic regression has some advantages compared to other methods.

5.4 Unsupervised Learning

The term unsupervised learning refers to analysis goals without training data for the evaluation of the analysis results. In this section, we focus on unsupervised learning problems which aim at a segmentation of the data. They are summarized under the heading cluster analysis. From the numerous approaches to cluster analysis, we consider in this section hierarchical methods, partitioning methods, and model-based clustering. Clustering for temporal data will be treated in Chap. 6 and cluster analysis for text mining in Chap. 8.

5.4.1 Introduction and Terminology

In the matter of unsupervised learning, our data structure is similar to supervised learning, but we have only observed input variables $X = (X_1, X_2, \ldots, X_p)$ and no observed output variable. The analytical goal is finding a grouping of the observations, so-called *clusters*, that can be used later on for explaining the structure of the observations in the context of the domain. We will discuss two different modeling approaches for defining the clusters: *distance-based methods* and *model-based methods*.

In the first case, the main prerequisite for modeling is the definition of a distance between observations, which measures the similarity or dissimilarity of observations. In the case of quantitative variables, the most important distance is the Euclidean distance defined for two p-dimensional vectors x and z by the equation

$$d^2(x,z) = \sum_{j=1}^{p} (x_j - z_j)^2 = ||x - z||^2. \tag{5.32}$$

In the case of qualitative variables, the most frequently used distance is the *Hamming distance*. This distance uses dummy coding for the different values of the qualitative variable, i.e., each value corresponds to a dummy variable with the

values 0 and 1. Using such coding, we obtain a bit string for each observation, and the distance between bit strings is defined by the number of different bits. Note that this distance corresponds to the Euclidean distance for the dummy variables.

Measuring the distance between observations with variables of different magnitude puts an emphasis on the distance between the variables with larger scale. Hence, it is usually recommended to standardize the variables into a value range [0, 1]. Such standardization is of utmost importance if we want to measure the distances between observations with qualitative and quantitative variables. A detailed description of a general method for defining distances can be found in [9] and is implemented in R[1] as the distance function *daisy*.

Two different approaches are commonly used for clustering observations based on distances. The first one comprises so-called *hierarchical methods* which define a hierarchy tree for the observations. Each node in the tree represents a possible subset (cluster) of the observations, the root defines one cluster containing all observations, and the leafs of the tree are the observations representing N different clusters. Cutting the tree at a certain level provides a possible cluster solution. We will treat such methods in Sect. 5.4.2. The second approach consists of so-called *partitioning methods* which define a number of clusters in advance and assign the observations iteratively to the clusters. Such methods will be treated in Sect. 5.4.3.

Besides the formulation of clustering based on distances, we can formulate the problem in the context of finding the distribution of the observations as described in [12] under the topic *descriptive methods*. Such methods assume that the distribution of the observations is a mixture distribution:

$$f(x) = \alpha_1 f_1(x) + \alpha_2 f_2(x) + \alpha_p f_p(x). \tag{5.33}$$

Here, p is the number of clusters, α_i are the proportions of the clusters in the observations, and $f_i(x)$ are the densities of the distributions in each cluster. The number of clusters p, the mixture probabilities α_i, $\sum \alpha_i = 1$, and the distribution in the components are unknown. In Sect. 5.4.4, we will present a solution for this approach under the assumption that the distribution in the different clusters is a multivariate normal distribution.

As soon as we have defined a cluster solution, we are also interested in the representation of the clusters by a typical element of the cluster. This characterization connects cluster analysis with the method of *vector quantization*, which is a more decision-theoretic approach for dimensionality reduction. The goal in vector quantization is finding p representatives for the clusters, for example, a customer with typical behavior. Details on the difference between vector quantization and clustering may be found in [4]. The main difference to vector quantization is that clustering does not use a decision-theoretic-oriented criterion for the overall evaluation of a solution, but more application-oriented criteria, such as reliability and validity, as discussed in 2.1.4.

[1] http://www.r-project.org/.

The validity and reliability of the solution are also of utmost importance for model assessment, model selection, and model evaluation. Besides this, we will consider a number of diagnostic tools for model assessment. With respect to model testing, the method of using an independent set of test data is usually recommended. The following template summarizes the main steps in cluster analysis:

Template: Cluster Analysis

- **Relevant Business and Data:** Customer behavior represented as cross-sectional data for process instances with a matrix **X** of explanatory variables
- **Analytical Goals:**
  - Find a segmentation of the data into clusters which allows an interpretation from domain point of view
  - Determine representatives for the clusters
- **Modeling Tasks:** Definition of a model for data description either based on the distances between the observations or by a mixture model for the distribution
- **Analysis Tasks:**
  - *Splitting Data:* If necessary split the data randomly into one set for training and another for testing the model
  - *Model Estimation:* Estimate the cluster solutions
  - *Model Assessment:* Assess the quality of models with respect to homogeneity of the clusters, separation between clusters, validity and reliability
  - *Model Selection:* Select a model by specifying the number of clusters
- **Evaluation and Reporting Task:** Evaluate the selected model using test data

5.4.2 Hierarchical Clustering

Starting from a distance matrix between observations, we can define a distance $D(C_j, C_k)$ between sets C_j and C_k representing clusters of observations, called *linkage*. Figure 5.15 shows two frequently used techniques, i.e., the *average linkage* in the right panel and *complete linkage* in the left panel. Average linkage is defined as the mean distance between all observations in the two clusters and complete linkage as the maximum of the distances between points in the two clusters. A third important technique is the Ward method, which measures the distance between two clusters by comparing the total within-cluster sum of squares for the two clusters separately with the within-cluster sum of squares resulting from merging the two clusters.

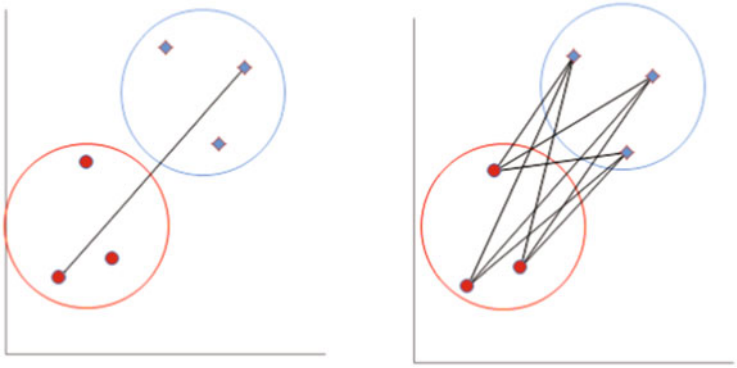

Fig. 5.15 Distances between clusters using average linkage (*left*) and complete linkage (*right*)

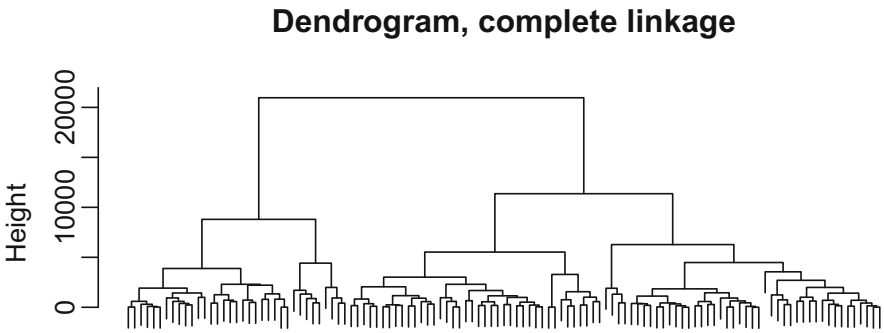

Fig. 5.16 Cluster tree with complete linkage (R package`cluster`)

Based on a definition of the distance between clusters, one can construct a cluster tree using *agglomerative methods* as outlined in the following algorithm:

Algorithm 4: Agglomerative clustering

```
Define clusters C_k, 1 ≤ k ≤ N by the observations, N_cl = N;
for k = 1 to N − 1 do
    Merge clusters C_r and C_s for which d(C_r, C_s) = min_(l,k) D((C_l, C_k));
    N_cl = N_cl − 1;
end
```

The resulting hierarchy can be visualized by using a tree diagram or *dendrogram* as shown in Fig. 5.16. The length of the branches called height corresponds to the distance between the merged objects.

Using the tree, one can decide about the number of clusters according to the height. Merging two clusters is not advisable if the height of the branches measured from the merged cluster to the individual clusters is a substantial value. The evaluation of distances is easier using a *scree plot*, which shows the distance between the clusters plotted against the number of clusters. Figure 5.17 shows the

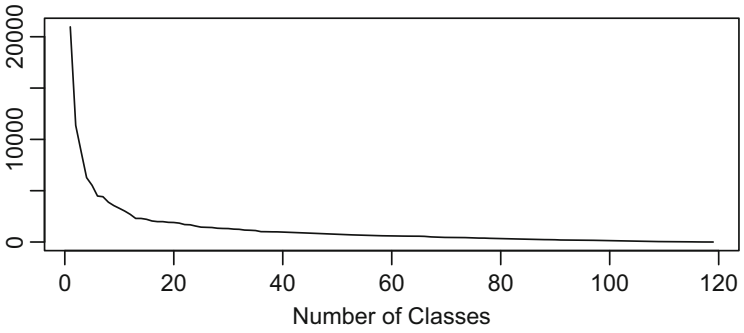

Fig. 5.17 Scree plot for cluster fusion (R graphics)

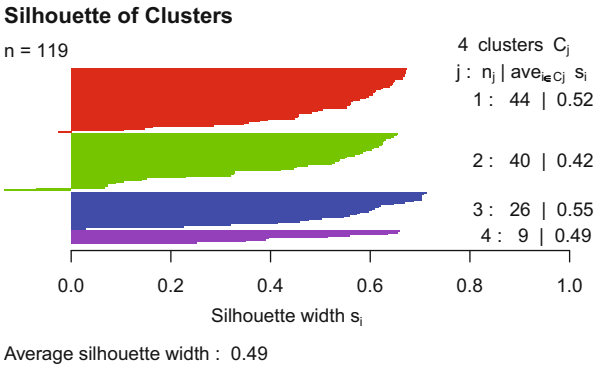

Fig. 5.18 Silhouette plot for cluster solution (R package `cluster`)

scree plot for the cluster tree depicted in Fig. 5.16. As one can learn from both figures, a solution with four clusters seems reasonable.

As already mentioned in Sect. 5.3.1, the evaluation of a cluster has to be mainly accomplished in the context of the domain problem. The interpretation of the clusters can be supported by descriptive statistics for variables. A formal evaluation of quality of the cluster assignment can be done using *silhouettes*. For constructing the silhouette, two different distances are calculated: the average distance from each observation to the observations in the same cluster and the minimum distance from the observation to observations in other clusters. From these two distances, silhouette values for each observation are calculated by standardization into the interval $[-1, 1]$. The silhouette values of the observations are plotted for each cluster in decreasing order as shown in Fig. 5.18. Large positive values indicate that the observation is well located in the cluster. Negative values are an indicator that the observation may not be well assigned. Besides this visual inspection of the cluster solution, one can calculate a silhouette coefficient as the overall quality measure of the solution. Details may be found in [18].

In the following, we will demonstrate the principle of clustering in the context of the HEP use case (cf. Sect. 1.4.2).

HEP Use Case: Clustering of Student Performance

In practical exercises about algorithms, students have to solve programming examples which are uploaded and tested. The following data were collected from 109 students attending the course: the phase of the course in which the solution was uploaded; the number of trials the student made; the size of the program; the minimum, maximum, and average length of the messages written in the forum; and the number of read-and-write activities in the forum. Cluster analysis was done with and without standardization of the variables. For not standardized variables, the cluster tree is shown in Fig. 5.16. The dendrogram and the scree plot in Fig. 5.17 suggest a solution with four clusters. Due to the fact that the variable size of the program has a larger scale than the other variables, the division into the clusters is dominated by the variable size. The interpretation of the clusters is as follows:

- The first cluster comprises 44 students with short programs uploaded in the first phases. In average, these students needed only 14 trials. With respect to the forum activities, these student, were not very active.
- The second cluster consists of 40 students who uploaded longer programs, needed about 30 trials in average, and were more active in the forum.
- The third cluster of 26 students uploaded longer programs than the first cluster, and they were much more active in writing in the forum.
- Finally, the fourth cluster comprises nine students with very large programs, which were uploaded very late, after almost 40 trials in average.

Figure 5.18 shows the silhouette plot of the solution. The silhouette coefficient is 0.49.

As expected, the analysis of the standardized data showed that the size of the program does not dominate the classification. More important are the activities in the forum. This solution also shows two clusters which are dominant in size, and two clusters encompass the students with exceptional behavior.

This example shows that both approaches provide results which can be interpreted from a subject matter point of view.

Details can be found on the homepage of the book:

www.businessintelligence-fundamentals.com

5.4.3 *Partitioning Methods*

Partitioning methods define a cluster solution for observations for a given number of clusters in an iterative way. The most popular method is *K-means* which defines the solution by the following algorithm:

Algorithm 5: K-means algorithm

Data: Observation matrix **X** and distance for the objects; number of clusters K.

Result: Cluster solution for observations

1 **begin**

2 Define an initial solution for the cluster centers $(c_1, c_2, \ldots, c_k)$;

3 Assign each observation x to the cluster which center is closest to the observation;

4 Compute new centers for the clusters as means of the assigned observations;

5 Repeat steps 2 and 3 as long as there is no significant change in the centers;

6 **end**

Two decisions of the user are important for the application. The first one is deciding about the number of clusters. For this decision, it is useful to plot the sums of squares within the clusters for different solutions and use this diagram in a similar way as the scree plot in the case of hierarchical clustering. Figure 5.19 shows such a graph for solutions between one and ten clusters. Further, a solution with four or five clusters seems to be a reasonable choice in this case. The second decision is about the initial centers. The standard approach is to choose the centers at random and to compute different solutions. In most implementations, one can also provide starting solutions.

The simple structure of the algorithm makes K-means an attractive procedure for large data sets, and it can be implemented easily on parallel architectures. The iterative structure also allows modifications for applications with rapidly changing

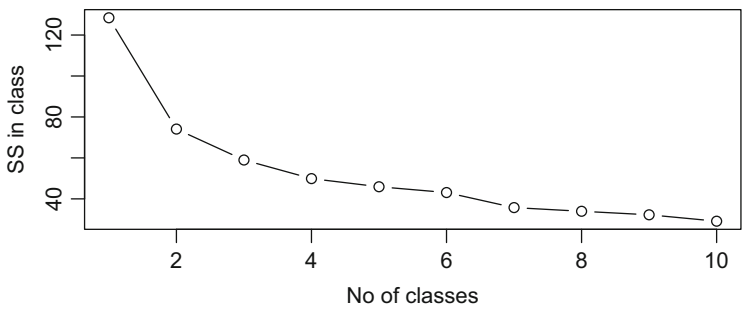

Fig. 5.19 Sum of squares within clusters (R graphics)

data from transactional databases. K-means is also frequently used for vector quantization.

One disadvantage of the procedure is the lack of robustness with respect to outliers. In the case of clustering, outliers can be interpreted as small groups of observation with rather irregular behavior. Such data may cause solutions which are rather sensitive regarding the starting values.

A more robust alternative to K-means is the *PAM algorithm*, which stands for partition around medoids. The concept of a *medoid* is a multidimensional generalization of the concept of the median, i.e., a central point in the multivariate data. Contrary to the mean, a medoid is not computed by a formula, but it is an existing data point. For a precise definition, we refer to [18]. The algorithm also works iteratively from a starting set of medoids. One iteration consists of the assignment, the swapping, and the calculation of new medoids. In the assignment step, observations are allocated to the closest medoid. The swapping step tries to find observations in the clusters with smaller distances from the data points in the cluster than the existing medoid. The new medoids are defined by the points with the smallest distance from the actual solution. For details, consider [18].

In the following, we will demonstrate the application of both methods in the context of the CRM use case (cf. Sect. 1.4.4).

`CRM Use Case: Clustering of Customers`

We are interested in finding user profiles of customers with respect to the different services. Four services are of main interest: `Service 2`, `Service 3`, `Service 4`, and `Service 9`. Furthermore, the variables `Sales` for the actual overall sales, the activity index `ActInd` as measure for actual performance, and the variable `SalesInd` for sales in the past are used. A first data inspection showed that the variables have numerous outliers, mainly due to the fact that there are extreme power users. Hence, we decided to perform two different analyses. The first one used the data of 528 regular users, where K-means was used for standardized data. Figure 5.19 shows the decrease in sum of squares for solutions with 1 up to 10 clusters. We chose a solution with five clusters. The 528 observations are distributed to the clusters as follows:

Cluster 1, 43 %; Cluster 2, 4 %; Cluster 3, 25 %; Cluster 4, 14 %; and Cluster 5, 14 %.

For the interpretation of the cluster, we used the distribution of the variables in the clusters shown in Fig. 5.20 for `Sales`, `Service 2`, and `Service 3`. This leads to the following interpretation:

- Cluster 1 may be characterized as "small users."
- Cluster 2 is a small group of intensive users, who are the main users of `Service 2`.
- Cluster 3 are moderate users using mainly `Service 3`.
- Cluster 4 is a group of heavy users, with high usage in `Service 3`.
- Cluster 5 are average users, mainly using `Service 3`.

Fig. 5.20 Distribution of variables within clusters (R graphics)

For the second data set with 1,311 users, K-means was not able to calculate a useful solution. Although we used standardized variables in this case, a solution with five clusters showed one major cluster comprising more than 75 % of all cases of average users and four clusters differentiating between different types of extreme power users. This problem was independent from the chosen starting values. The PAM algorithm produced a more reliable solution. It differentiated between 519 low-level customers using mainly `Service 2` and `Service 3`, 490 average customers using mainly `Service 3`, a group of 199 heavy users with a high activity index and high usage of `Service 3`, and two small groups of 53 and 50 customers being power users, one mainly in `Service 2` and the other mainly in `Service 3`.

More results can be found on the homepage of the book:
www.businessintelligence-fundamentals.com

Besides these two clustering methods, one can find many other clustering algorithms in the literature. A well-known and frequently used method are *self-organizing maps* (SOM), which define a neural network for clustering. In [14], it is shown that SOMs can be also interpreted as a special kind of K-means clustering, where the clusters are defined by a distorted grid. In IBM/SPSS®[2] two-stage clustering is used, which is a combination of hierarchical methods and K-means clustering based on the idea of a cluster feature tree. Details may be found in [26].

5.4.4 Model-Based Clustering

Model-based clustering reformulates the problem of finding groups into a problem of estimating a mixture model for the distribution of the observations. We will consider here only the case of mixtures of normal distributions. The model starts

[2] http://www-01.ibm.com/software/analytics/spss/.

from the assumption that observed data $\mathbf{X}$ are generated from a population defined by p subpopulations $G_1, G_2, \ldots, G_p$. For the clusters in the population, the term strata is frequently used. The proportion of subpopulation G_j in the population is denoted by α_j. In each subpopulation, the observed variables are multivariate normally distributed with mean vectors μ_j and covariance matrices Σ_j. Under these assumptions, we can denote the density function of an observation x as a mixture of normal distributions

$$f(x) = \sum_{j=1}^{k} \alpha_j \phi(\mu_j, \Sigma_j), \quad \alpha_j \geq 0, \ \sum_{j=1}^{k} \alpha_j = 1. \tag{5.34}$$

This model transforms the cluster problem into the problem of estimating the unknown parameters μ_j and Σ_j of the normal distributions and the proportions α_j. If we know the parameters of the model, we can determine for an observation x_i the values of the densities $f_j(x)$ and assign the observation to the most plausible cluster defined by the group with the highest value for the density. Note the similarity to a classification problem.

The major challenge of this approach is that we have no training data for learning the probabilities, and we do not know in advance how many different clusters exist. The solution is achieved in two steps similar to the ideas for model selection in supervised learning. In a first step, the parameters of the model are estimated given a fixed number of clusters, and in a second step, a model is selected according to a model selection criterion as introduced in Sect. 5.2.2.

For the estimation of the parameters μ_j, Σ_j, and α_j for a given number of clusters, the *EM algorithm* is used. This algorithm is a general method for computing maximum likelihood estimates in case of missing information. We will explain the basic idea of the algorithm only for the case of a mixture of two normal distributions with the same covariance structure. A general treatment of the EM algorithm may be found, for example, in [25] or [14].

In the case of a mixture of two normal distributions with the same covariance structure, the density of the observations is defined by

$$f(x, \theta) = \alpha\phi(x, \mu_1, \Sigma) + (1 - \alpha)\phi(x, \mu_2, \Sigma), \ \theta = (\alpha, \mu_1, \mu_2, \Sigma) \tag{5.35}$$

For the group membership of the observation, we introduce a dummy variable Z

$$Z(X) = \begin{cases} 1 & \text{if } X \in \text{Group } G_1 \\ 0 & \text{if } X \in \text{Group } G_2 \end{cases} \tag{5.36}$$

for class membership. Using this dummy variable, we can write the distribution of an observation as

$$f(x, z, \theta) = \alpha^z \phi(x, \mu_1, \Sigma)^z + (1 - \alpha)^{1-z} \phi(x, \mu_2, \Sigma)^{1-z}, \ \theta = (\alpha, \mu_1, \mu_2, \Sigma) \tag{5.37}$$

and understand the distribution in (5.32) as the marginal distribution of the variables (X, Z) with respect to X. If we would know the values of Z for each observation, the estimation of means μ_j and covariances Σ_j could be done by standard formulas. The EM algorithm uses this fact and defines the solution in two steps. The basic iteration is shown in the following algorithm:

Algorithm 6: EM algorithm

1 **repeat**

2 Expectation-Step: Given estimates $\hat{\theta} = (\widehat{\mu_1}^{(r)}, \widehat{\mu_2}^{(r)}, \hat{\Sigma}^{(r)}, \hat{\alpha}^{(r)}))$ for the parameters, compute the expected values of the loglikelihood given the data and the parameter estimate:
$J(\theta, \hat{\theta}^{(r)}) = E[\log(\frac{f(x,z,\theta)}{f(x,z,\hat{\theta}^{(r)})}|(x, \hat{\theta}^{(r)})]$;

3 Maximization-Step: Find new parameter values
$\hat{\theta}^{(r+1)} = \arg\max_{(\theta)} J(\theta, \hat{\theta}^{(r)})$;

4 **until** *convergence is reached*;

The idea behind the expectation step is replacing the unknown variable Z by the expected value. In the case of normal distributions, the calculation can be done using Bayes' theorem. It can be shown that this procedure converges, although the iteration could be rather slow.

In model selection, these calculations are done for different numbers of clusters and for different possible assumptions about the covariance structure for the observations. These covariance may be the same in all groups, but they may also differ between the classes. For the selection of a model, the BIC criterion (cf. Sect. 5.2.2) is used. The chosen model has the smallest value for BIC.

5.4.5 Summary: Unsupervised Learning

One important goal in unsupervised learning is finding clusters of similar observations. The primary modeling task in cluster analysis is the definition of the similarity, or distance, between the observations using the observed variables. The most frequently used distance for quantitative variables is the Euclidean distance; in the case of qualitative variables, one can use the Hamming distance after dichotomization of the variables by dummy variables. An important decision in modeling is whether the variables should be standardized. In general, standardization is preferred. Based on the distances, two basic analysis techniques can be used. The first one are hierarchical methods which define clusters by aggregation procedures. Important decisions for these algorithms is the definition of the distance between the clusters and the number of clusters. The second approach are partitioning methods which start with a predefined number of clusters and assign iteratively the observations to the clusters.

The model selection task in cluster analysis is the determination of the number of clusters. Various descriptive tools can be used which support the selection of the number of clusters. Also for the evaluation of the cluster solution, these descriptive tools can be applied. For assessment of the generalization of the cluster solution, splitting the observations in training data and test data is appropriate.

Besides these two basic techniques, there are numerous algorithms which combine the two approaches. An alternative to distance-based methods is using a model-oriented approach that defines a mixture model for the observations. The estimation of the parameters of the mixture model uses the EM algorithm.

5.5 Conclusion and Lessons Learned

In this chapter, we introduced different techniques for the analysis of cross-sectional data. Depending on the analysis goal, we distinguished between methods for the achievement of predictive goals and for descriptive goals. Predictive goals are formulated as regression problems or as classification problems. In both cases, the definition of an appropriate loss function is essential. From the methodological point of view, balancing between overfitting and underfitting has to be done using a training and a test sample. In the case of unsupervised learning, we considered the formulation as clustering problem based on distances and as estimation problem for mixture distribution.

5.6 Recommended Reading

There exist many excellent books about data mining and machine learning which cover the material of this chapter in more detail. An introduction which emphasizes business applications is Linoff and Berry (2011) ([19]). For a more theoretical oriented approach from the statistical perspective, we recommend Hastie et al. (2009). A concise description of the most important data mining algorithms can be found in Wu and Kumar (2009). For readers interested in the algorithmic perspective and using Python for data mining, we recommend Marsland (2009).

- Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning. Springer, 2nd edition
- Linoff GS, Berry MJA (2011) Data mining techniques for marketing, sales, and customer relationship management. Wiley
- Marsland S (2009) Machine learning—an algorithmic perspective. CRC Press
- Wu X, Kumar V (2009) The top ten algorithms in data mining. CRC Press

References

1. Bowman AJ, Azzalini A (1997) Applied smoothing techniques for data analysis. Oxford Science Publications, Oxford
2. Breiman L, Friedman J, Stone CJ, Olshen RA (1984) Classification and regression trees. CRC, Boca Raton
3. Burnham KB, Anderson DA (2004) Model selection and multimodel inference: a practical information-theoretic approach. Springer, New York
4. Cherkassky V, Mulier F (2007) Learning from data—concepts, theory and methods. Wiley, New York
5. de Boor C (2001) A practical guide to splines. Applied mathematical sciences. Springer, New York
6. Efron B, Tibshirani RJ (1993) An introduction to the bootstrap. Chapman & Hall/CRC, Boca Raton
7. Faraway JJ (2004) Linear models with R. Chapman & Hall/CRC texts in statistics. CRC Press, Boca Raton, FL
8. Friendly M, Kwan E (2009) Where's Waldo? Visualizing collinearity diagnostics. Am Stat 63(1):56–65
9. Gower JC (1971) A general coefficient of similarity and some of its properties. Biometrics 27:857–874
10. Günther F, Fritsch S (2010) Neuralnet: training of neural networks. R J 2(1):30–38
11. Hand DJ, Yu K (2001) Idiot's Bayes—not so stupid after all? Int Stat Rev 69:385–398
12. Hand DJ, Mannila H, Smyth P (2001) Principles of data mining. MIT, Cambridge, MA/London
13. Hastie TJ, Tibshirani RJ (1990) Generalized additive models. Chapman & Hall/CRC, Boca Raton
14. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning, 2nd edn. Springer, New York
15. Haykin S (2008) Neural networks and learning machines. Prentice Hall, Upper Saddle River
16. Hornik K, Stichcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Netw 2:359–399
17. Hosmer DW, Lemeshow S (2000) Applied logistic regression. Wiley texts in statistics. Wiley, New York
18. Kaufman L, Rousseeuw PJ (1990) Finding groups in data: an introduction to cluster analysis. Wiley-Interscience, New York
19. Linoff GS, Berry MJA (2011) Data mining techniques for marketing, sales, and customer relationship management. Wiley, New York
20. Maronna RR, Martin D, Yohai V (2006) Robust statistics—theory and methods. Wiley, New York
21. Marsland S (2009) Machine learning—an algorithmic perspective. CRC, Boca Raton
22. Ramsay JO, Silverman BW (2005) Functional data analysis, 2nd edn. Springer, New York
23. Therneau TM, Atkinson EJ (2014) An introduction to recursive partitioning using RPART routines. http://cran.r-project.org/web/packages/rpart/vignettes/longintro.pdf. Accessed 24 June 2014
24. Vert M, Tsuda K, Schölkopf (2004) A primer on kernel methods. In: Kernel methods in computational biology. MIT, Cambridge, pp 35–70
25. Wu X, Kumar V (2009) The top ten algorithms in data mining. CRC, Boca Raton
26. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: Jagadish HV, Mumick IS (eds) SIGMOD'96: International conference on management of data. ACM, New York, pp 103–114