Multiple Testing Error Assessment for Strongly Dependent Hypotheses using Resampling

Diploma Thesis

Otto-von-Guericke-Universität Magdeburg
Institute of Knowledge and Language Engineering
Prof. R. Kruse

Otto-von-Guericke-Universität Magdeburg
Institute for Mathematical Stochastics
Prof. R. Schwabe

MelTec GmbH & Co. KG

applicant
Sebastian Nusser

supervisors
Dr. Christian Borgelt, Otto-von-Guericke-Universität
Dr. Peter Karcher, MelTec GmbH & Co. KG

starting date: 01.03.2005
ending date: 31.07.2005

Sebastian Nusser (mail@seb-nusser.de)
Multiple Testing Error Assessment for Strongly Dependent Hypotheses using Resampling
Diplomarbeit, Otto-von-Guericke-Universität Magdeburg
Magdeburg, 2005-07-28

Abstract

In this thesis a method is investigated to assess the error for the multiple testing problem that arises in data mining algorithms. This method is based on a resampling procedure and it can deal with strongly dependent hypotheses. This procedure aims at situations where the number of independent samples is small and it would be expensive to obtain more independent samples. The basic requirements and crucial aspects to use this approach are given. The applicability of this method is demonstrated in detail for the Apriori algorithm and for further data mining methods the general applicability is shown as well by the example of a decision tree learner. Furthermore an example for a biological application is given.

Contents

Chapter 1 – Introduction

This chapter gives a motivation in Section 1.1, why a method to control the multiple testing problem should be used and Section 1.2 gives a more detailed problem description for MelTec. The last section gives an overview of this thesis.

1.1 Motivation

"Die Suche nach Wahrheit ist köstlicher als deren gesicherter Besitz.[1]"
(Gotthold Ephraim Lessing)

In bio-sciences, as well as in many other scientific fields like geo-sciences and environmental sciences, large data sets are collected. Analyzing such data sets is often performed by standard data mining approaches. These approaches search through a large space of models and they perform certain tests to evaluate possible models. Although these tests are mostly no typical formal statistical tests, but they can be seen as such. As is well-known from statistics, any statistical test has a certain probability of failure (see Section 2.2). Thus, for the tests in a data mining approach, there also exists a probability of failure – but often this probability cannot be controlled directly. Furthermore, even if the probability of failure can be determined for each single test, there is no statement about the quality of the whole result, which consists (in fact) of many tests. This increases the uncertainty about the quality of the result of a data mining approach.

> For example: When 100 tests are performed and for each test there is a probability of failure of five percent, the probability of failure over all tests is $(1 - (1 - 0.05)^{100}) = 0.9941$. Thus, it becomes quite sure that at least one of the results, which are returned by the data mining algorithm, is false.

There are statistical methods (see Section 4.2) to control such a probability over all tests – but in the field of computer sciences, the problem of controlling the quality of the results in such a statistical way is not very common. Since data mining algorithms perform several thousands of tests and each test has an unknown probability of failure, no direct control of the quality of the whole result is possible. Our approach is slightly different than the standard statistical approaches to control the probability of failure. It uses the concept of permutation tests (see Section 2.3) to provide a suitable test statistic. This test procedure allows us to determine the error rate under

1 The search for the truth is more delicious than knowing the truth for sure.

the assumption that the data contains no information and the result of the data mining algorithm consists only of random events. Furthermore, using a statistical description of the quality of a result can enhance the understanding of a problem for biologists and other scientists, since statistical descriptions are quite common in their scientific community. Another advantageous feature of our approach is that there is no need to adapt the data mining algorithm to work with our approach, so it becomes easy to evaluate different data mining approaches and to choose the most suitable.

1.2 Problem Description for MelTec

MelTec is a bio-science company, which is specialized in the research of intracellular protein expressions and interactions. For research purposes large data sets are generated from biological samples. Unfortunately, the number of independent samples is often quite small, since each independent sample must be taken from a different patient and deriving further samples can be expensive in time and costs. The small number of independent samples reduces the certainty that the results of the data mining process contain useful information, which can distinguish different groups of patients. So it is necessary to determine whether the result of the data mining process was a random event or whether there is a certain unknown model behind the data. Since the typical research process in bio-sciences consists of repeated experiments to validate the results which are obtained in previous experiments, it is not necessary to determine the quality of each single result of the result set. But it is important to know whether there exist useful results in the result set. Thus, the result set should be investigated by further research. Exactly for this purpose our approach is designed. It determines whether further experiments (which could be very expensive – especially in the field of medical and bio-scientific research) would make sense – or not.

1.3 Overview

In Chapter 2 the basics of statistical testing are reviewed and the hypothesis testing procedure is explained in detail – this part is essential to understand the underlying statistical theory of our approach. Furthermore, this chapter introduces the concept of permutation tests. The second basic field of theory for this work is given in Chapter 3. This chapter gives a short introduction into the field of data mining algorithms and it explains in more detail two popular data mining approaches, which are used for our experiments. In Chapter 4 the multiple testing problem is explained in a more formal way and common approaches to solve this problem are discussed briefly – the multiple testing problem is the crucial aspect for developing our framework. Our framework, its theoretical background, and its basic algorithm are given in Chapter 5. The experiments, which are performed with several data sets, and their results are discussed in Chapter 6; including technical details of the implementation, which is used to perform the experiments. A brief description of the testing data and the results of the analysis of this approach is also given in this chapter. The application of the method to real world data in the field of bio-science is given in Chapter 7. This experiment is an example for the biological application for MelTec. In this chapter MelTec's technology is introduced and the basic information of the adaptation of our approach to the MELK technology are given. The last chapter discusses our approach and gives possible steps that could be seen as further work.

Chapter 2 – Statistical Testing

This chapter introduces briefly the basics of statistical testing. These basics are the theoretical basis of our approach. In Section 2.1 basic terms and definitions for statistical testing are reviewed. Section 2.2 introduces in more detail the concept of hypothesis testing and Section 2.3 reviews the concept of permutation tests. For more details about the field of statistical testing look at [Milton1990] and for resampling procedures see, for example, [Manly2001] and [Efron1994].

2.1 Basic Terms and Definitions

In the field of statistics there are some commonly used terms, which are introduced in the following.

The process of observing the outcome of a chance experiment is called a **random experiment**. The **elementary outcomes** are all possible results of the random experiment and an **event** is a statement about the outcome of the experiment. The **sample space** is the collection of all elementary outcomes of the random experiment.

The **probability** measures the likelihood of the occurring of the elementary outcomes. It assigns to each elementary outcome a value larger or equal than zero and the total probability of all elementary outcomes is one. The probability of the union of several disjoint events is equal to the sum of the probabilities of each single event. The **distribution** is a function, which assigns to each event a certain probability.

The entire group about which inferences is made will be called a **population**. Since the size of a population can be infinite or at least very large, it is often not possible to study the population in its entirety. Commonly, there is no information about the characteristics of the population known.

A subset of the population will be called a **sample**. It is used to represent the entire population. A sample can be studied in detail and information from the sample are used as a basis for statistical testing. The sample's distribution and its characteristics can be observed. Often, it is assumed that the members of the sample are drawn randomly and independently from the population.

A function of the sample data is called a **statistic**. It is used to estimate or represent unknown **parameters** (unknown population values or characteristics) of the population. Such a value which is calculated from sample data and which is used to test a hypothesis (see Section 2.2) is called a **test statistic**.

The actual (and unknown) distribution of values of any variable for the entire population is called **population distribution** and the actual (and known) distribution of values on any variable for the sample is called **sample distribution**. The **sampling distribution** describes probabilities associated with a statistic when a random sample is drawn from a population.

2.2 Hypothesis Testing

A statistical statement about the parameters of one or more populations is called a **hypothesis**. The statistical procedure to make a decision between two contradicting hypotheses about the process that generated a certain data set is called **hypothesis testing**. The first hypothesis is called the **null hypothesis** and denoted H_0. The second hypothesis is called **alternative hypothesis** and denoted H_A. The alternative hypothesis will be accepted, if the observed data values are sufficiently unlikely under the assumption that the null hypothesis is true. Otherwise, the null hypothesis is not rejected[2]. But not rejecting the null hypothesis is not equivalent to accepting the null hypothesis.

> For example: If a hypothesis is stated as "the mean age of a student is equal to 23". There are three different ways to state the null hypothesis and the alternative hypothesis: (1) H_0 is stated as "the mean age of a student is equal to 23" and H_A is stated as "the mean age of a student is not equal to 23", (2) H_0 is stated as "the mean age of a student is larger than or equal to 23" and H_A is stated as "the mean age of a student is smaller than 23", and (3) H_0 is stated as "the mean age of a student is smaller than or equal to 23" and H_A is stated as "the mean age of a student is larger than 23".

The decision to reject the null hypothesis is made by observing the value of some statistic S whose probability distribution can be determined under the assumption that s_0 is the true value of the population parameter s ($H_0 : s = s_0$). Such a statistic is called a **test statistic**.

The probability that the values of the test statistic S under the null hypothesis H_0 become more extreme than the observed value is called the **p-value**. The range of s can be divided into two regions: the **acceptance region** and the **critical region** (or rejection region). The critical region for a test represents the values of the test statistic that lead to a rejection of the null hypothesis. The probability that the observed value of the test statistic will fall into the critical region by chance if $s = s_0$ is called the **level of significance** α of the test. In a hypothesis testing study, α is the probability of committing a Type I error (see below). The p-value is the smallest level at which one could have a certain α and still have been able to reject H_0. So, the smaller the p-value, the smaller the probability of committing a Type I error will be.

Two possible types of errors can occur:

- Type I: the null hypothesis H_0 is rejected, even though it is correct.
- Type II: the null hypothesis H_0 is retained, even though it is false.

2 The preference of the null hypothesis is also called "benefit of the doubt" – only if strong evidence is against the null hypothesis the alternative hypothesis will accepted.

Due to the "benefit of the doubt" – the preference of the null hypothesis – the Type I error becomes more severe than the Type II error. Hence one wants to control the probability of committing the Type I error. This control is provided by the level of significance α.

The probability of making the correct decision of rejecting H_0 when H_A is true is called the **power** of the test.

The error types can be summarized by the following table.

Reality	**Retain H_0**	**Reject H_0**
H_0 is true	correct decision	Type I error
H_0 is false	Type II error	correct decision

Depending on the kind of the alternative hypothesis H_A (remember the example above), there are two categories of testing: **one-sided** and **two-sided testing**. The one-sided testing can be further divided as lower and upper one-sided testing.

(1) If $H_A : s \neq s_0$, the testing is two-sided. (The null hypothesis is $H_0 : s = s_0$.)

(2) If $H_A : s < s_0$, the testing is lower one-sided. (The null hypothesis is $H_0 : s \geq s_0$.)

(3) If $H_A : s > s_0$, the testing is upper one-sided. (The null hypothesis is $H_0 : s \leq s_0$.)

In the case of a two-sided test, for both upper and lower bounds, the significance level α is usually divided evenly. Each side of the distribution has a $\alpha/2$ probability of error. In the case of an one-sided test, α is only located at the specified side of the distribution. This is illustrated in Figure 1, which shows the possible acceptance regions for the example from page 4.

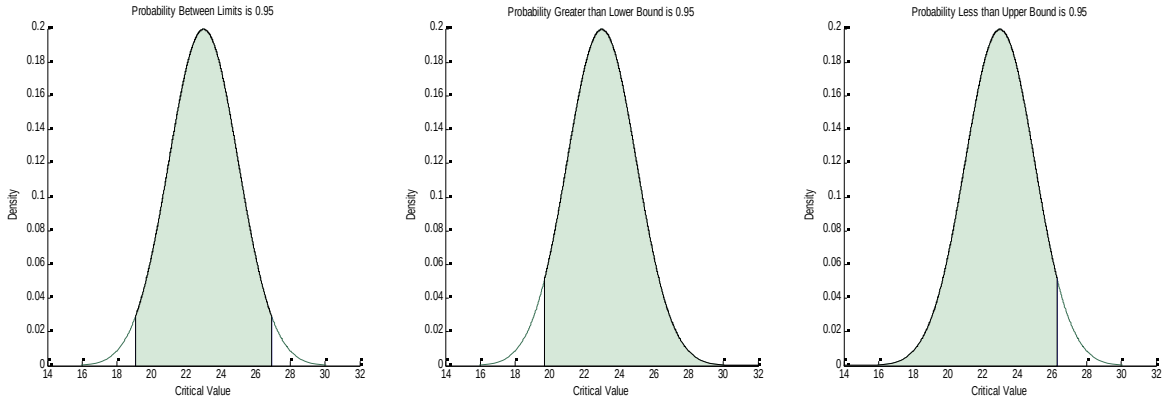


Figure 1: Two-sided testing (a), lower one-sided testing (b), and upper one-sided testing (c) for the example from above; where the shaded area is the acceptance region and a significance level α of 0.05 is used.

The following steps are used for hypothesis testing:

1. define the null hypothesis H_0
2. define the alternative hypothesis H_A, which is the hypothesis one would like to accept if sample data lead to rejection of H_0

3. identify test statistic S with a known probability distribution where H_0 is true
4. determine the critical region of the test statistic S based on distribution parameters, significance level α, and kind of alternative hypothesis
5. compute the value of the test statistic from the sample
6. draw conclusion
    - reject H_0 and accept H_A, if sample value is in critical region
    - retain H_0, otherwise

For common data mining methods, it is not possible to compute the exact distribution of the test statistic for step 3. Thus, this distribution must be estimated by a resampling technique like the permutation test (see Section 2.3).

2.3 Permutation Tests

Commonly one expects that there is a tendency for a certain type of pattern in a given data set. A permutation test is a procedure that compares an observed test statistic with a distribution that is generated by randomly reordering the data values. It is a way of determining whether the null hypothesis ("there is no pattern") is reasonable. A statistic S is chosen to measure the strength of a pattern in a data set. The distribution of S is obtained by randomly reordering the data. The observed value s of S is compared with the distribution of S as in Section 2.2.

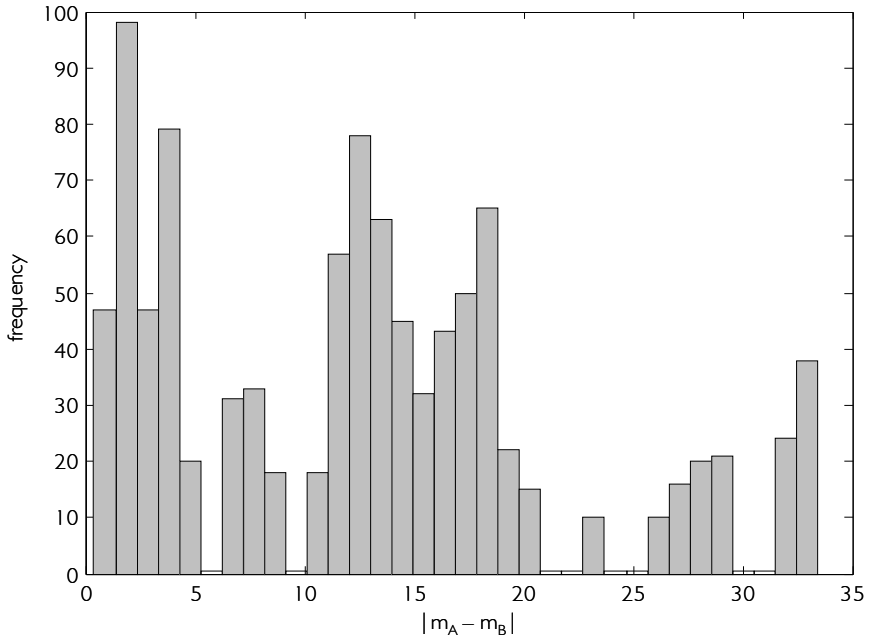


Figure 2: Resampling histogram for A = {13, 34, 43}, B = {44, 63, 69, 71, 73}, and k = 1000.

Consider, as an example, that the outcome of an experiment consists of two different sets $A = \{13, 34, 43\}$ and $B = \{44, 63, 69, 71, 73\}$. As statistic we will use the absolute difference between the arithmetical mean values of set A and set B. The means are $m_A = 30$ and $m_B = 64$, so the absolute difference between the means is $|m_A - m_B| = 34$. Now, the sets will be reordered randomly: $A^1 = \{43, 73, 34\}$ and $B^1 = \{44, 71, 13, 69, 63\}$, where $m_{A[1]} = 50$, $m_{B[1]} = 52$, and $|m_{A[1]} - m_{B[1]}| = 2$. By repeating this procedure k-times the resampling distribution as in Figure 2 can be derived.

The permutation test which is used in Section 5.3 is a common statistical procedure, which was first proposed by [Fisher1935]. The important assumption of this procedure is that the sample, which should be permuted, is not biased – that means the sample consists of independent observations, which are drawn evenly from the entire population. For a given sample set $\{(x_1,y_1), ..., (x_n,y_n)\}$ where x_i is the description of an object and y_i is the corresponding label, a sampling distribution is generated by reordering the y_i randomly.

Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a given data set, where x_i is the description of an object and y_i is the label of this object. Let σ be a function that reorders the values $\{1, \ldots, n\}$ randomly.

$\sigma : \{1, \ldots, n\} \rightarrow \{\sigma(1), \ldots, \sigma(n)\}$.

Then a suitable distribution for a permutation test can be derived by applying σ k-times on the labels of the objects, $\{(x_1, y_{\sigma(1)}), \ldots, (x_n, y_{\sigma(n)})\}$.

The resampling distribution is used to estimate the (in this case unknown) probability distribution under the null hypothesis as in step 3 of the general steps for hypothesis testing (see Section 2.2). The further steps for hypothesis testing remain the same.

Chapter 3 – Data Mining Algorithms

In this chapter a brief introduction to the field of data mining algorithms is given and two common approaches are explained in more detail – association rule mining in Section 3.2 and decision tree learning in Section 3.3. These approaches serve as examples for suitable data mining methods for our multiple testing error estimation method. Both data mining approaches, the Apriori and the decision tree learner algorithm, are used in our experiments in Chapter 6. For an overview over further data mining methods see, for example, [Mitchell1997] or [Wrobel2000].

3.1 Suitable Data Mining Algorithms

For our approach to estimate the multiple testing error there are some requirements, which the data mining algorithms should be able to handle. Since our approach tends in some sense to distinguish different groups, the data mining algorithm must provide results that allow to distinguish between these groups. It is necessary to define a so-called target attribute. The definition of a target attribute is essential for the permutation testing procedure, which is the main aspect of our approach, since the values of the target attribute will be changed randomly to derive the resampling distribution. Furthermore, it should be possible to distinguish results which contain information about the target attribute and results which do not describe the target attribute in some sense. As an example, if one regards all possible rules (and not only the rules with the target attribute), which are found by the Apriori algorithm, it could become very hard to discover any differences between the original data and the permuted data, since the amount of results will be huge in both cases and the influence of the target attribute becomes insignificant.

As commonly known, there are two different families of data mining algorithms – supervised and unsupervised. The goal for supervised learning tasks is to predict one single attribute based on the other attributes – this is also called predictive modeling. For unsupervised learning tasks, the goal is to discover patterns or segments of the data – this is also called descriptive modeling. For us it is not important whether the learner predicts the value of the target attribute, but it is important that its results, with regard to the target attribute, can be distinguished from other results.

Both families of data mining algorithms can be applied to our approach – as long as one can specify a target attribute which should be permuted and the result can be filtered for. For the family of unsupervised data mining algorithms the Apriori algorithm, which is explained in

Section 3.2, and for the family of supervised data mining algorithms the decision tree learning algorithms, which are explained in Section 3.3, serve as examples.

3.2 Association Rule Mining

Association rule mining ([Agrawal1993]) is one of the most popular approaches in data mining. This approach is used to discover regularities in given large databases. In the field of market basket analysis typical applications of association rule mining are catalog design, store layout, customer segmentation based on buying patterns, and so on. There are also other fields where this concept is used (for example in [BorgeIt2002]).

> For example: Consider an on-line shopping system where all transaction information (e.g. transaction date, package size, product name etc.) are stored in a database. This database can be used to optimize marketing programs and strategies, since one can extract information like: "If someone buys a digital camera and a photo printer it is likely that he will also buy some photo paper". With such information additional products can be offered to the customer, like "customers who bought this item also bought these ..."

Association rule mining consists of two steps: (1) determination of frequent item sets and (2) generation of rules from frequent item sets. The frequent item sets are sets that have at least a given minimum support. The support of an item set is the percentage of all transactions that contain the item set. To generate the association rules, for every frequent item set from the first step, all possible rules are generated and those with sufficient confidence are returned.

A more formal statement of the problem of association rule mining is given in [Agrawal1994]:

> Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items and let T be a set of transactions, where each transaction t is a set of items such that $t \subseteq I$. Let $s_{min} \in [0,1]$ be the minimal support and $c_{min} \in [0,1]$ the minimal confidence, which can both be chosen freely. A transaction t contains X, a set of items in I, if $X \subseteq t$. An association rule is an implication[3] of the form $X \rightarrow Y$, where $X \subseteq I$, $Y \subseteq I$, and $X \cap Y = \emptyset$.
>
> The rule $r : X \rightarrow Y$ holds in the transaction set T, if
>
> $$s(r) := \frac{|\{t \in T | X \cup Y \in t\}|}{|T|} \geqslant s_{min} \quad \text{and} \quad c(r) := \frac{|\{t \in T | X \cup Y \in t\}|}{|\{t \in T | X \in t\}|} \geqslant c_{min}$$
>
> holds.

Searching for frequent item sets can be viewed as traversing a tree as in Figure 3, where the possible subsets of size k are generated from the subsets of size k-1. For traversing such a tree there are two different approaches: a breadth first search like in the Apriori algorithm [Agrawal1993] and a depth first search like Eclat [Zaki1997]. Other algorithms can be found in [FIMI2003] and [FIMI2004]. To limit the search complexity: all sets that do not reach minimum

3 This kind of implication is not equivalent to a causal implication (like in [Pearl1991]).

support can be omitted (including their subtrees), because no superset of an infrequent item set can be frequent.

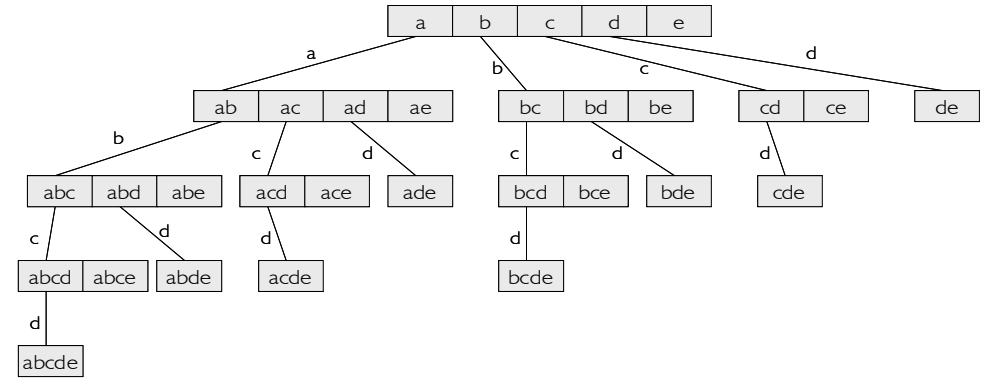


Figure 3: Search tree for five items a,b,c,d,e (from [Borgelt2002]).

In general, association rules are a descriptive and not a predictive method. However, by using attribute names in combination with the attribute values and by considering only rules with the target attribute, it is possible to use association rules for our method to estimate the multiple testing error of a data mining approach (for details see Section 6.1).

3.3 Decision Tree Learning

Another popular approach in the field of data mining are decision tree learning algorithms (e.g. [Quinlan1993]). Decision trees as classification method are quite simple to use. A decision tree is a tree-like structure of a set of tests and a set of classes. The inner nodes of a decision tree are attribute tests and the leaf nodes are assigned to class labels, see Figure 4. Leaves can be associated to the same class labels and inner nodes can be associated with the same attribute tests. A case is classified by starting at the root of the trees and following the attribute tests of the inner nodes until a leaf node is reached. This leaf states the class which is proposed for the particular case.

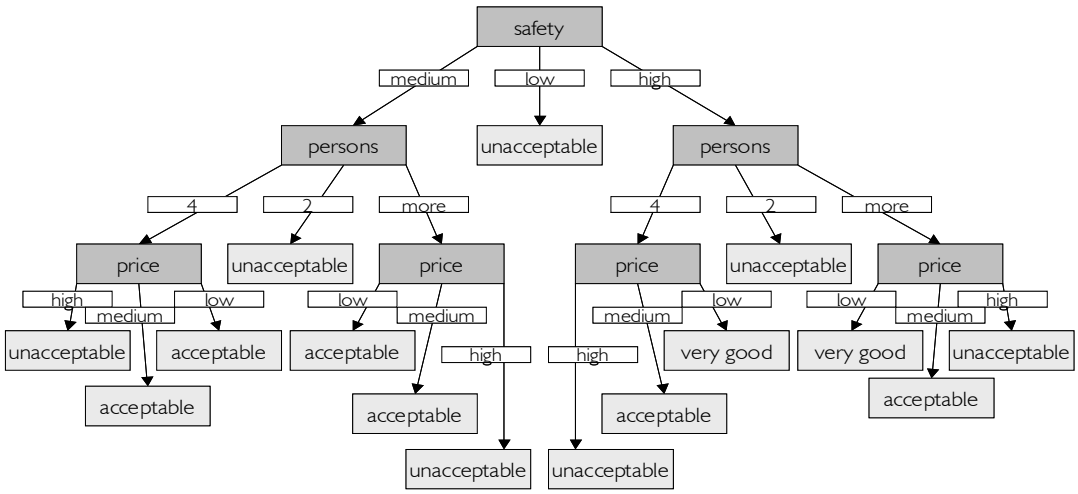


Figure 4: Decision tree for the conditions for renting a car for a family making holidays.

The decision tree in Figure 4 serves as an example. It can be used to find the appropriate rental car for a family making holidays. The criteria to decide on are safety level, number of seats, and rental fee. The possible class labels are "very good car", "acceptable car", and "unacceptable car". According to this tree, a car with a high safety level, which has more than 4 seats, and a medium price would be acceptable to rent for a family making holidays.

The usual approach to construct a decision tree is a top-down process. It is based on the "divide and conquer" principle. For a given data set compute a quality measures for all attribute tests. Select the attribute test with the best value of the quality measure. The example cases are divided according to the values of the test attribute. This procedure is applied recursively to the subsets until all cases belong to the same class or no more attribute tests are available.

For the top-down induction of decision trees (TDIDT) one can use the following recursive algorithm [Wrobel2000]:

Let E be the training set and Tests the set of all possible attribute tests for the given training set.

TDIDT(E,Tests)

- if all samples in E belong to one class or Tests is empty return leaf node with this single class or the most frequent class, otherwise:
- for each Test in Tests, determine Quality(Test,E)
- choose $Test_{max}$ which maximizes Quality(Test,E)
- split E into subsets $E_1,...,E_k$ according to $Test_{max}$
- for i = 1,...,k call T_i := TDIDT(E_i,Tests\{Test})
- return all subtrees $T_1,...,T_k$

A well-known quality measure is the information gain [Quinlan1993]. It measures how much information can be obtained by using a certain test attribute. The formal definition for the information gain as a quality measure is given, for example, in [Wrobel2000] as follows:

Let Test be a test that splits the training set E into the subsets $E_1,...,E_k$. Let $p_1, \ldots, p_m$ be the relative frequency of m classes in training set E and let $p_{i,1}, \ldots, p_{i,m}$ be the relative frequency of m classes in subset E_i. Then the information gain IG is determined by:

$$\text{Quality}(\text{Test},E) := \text{IG}(\text{Test},E) := I(p_1,\ldots,p_m) - \sum_{i=1}^{k} \frac{|E_i|}{|E|} I(p_{i,1},\ldots,p_{i,m}),$$

where $I(p_1,\ldots,p_m) = -\sum_{i=1}^{m} p_i \log(p_i)$ is the so-called Shannon Entropy.

A detailed discussion of other possible quality measures is given in [Borgelt1998].

Chapter 4 – Multiple Testing Problem

In this chapter the multiple testing problem is introduced and the importance of regarding it is shown. Section 4.1 describes this problem and introduces some basic terms. Some common approaches to deal with multiple testing problems are given in Section 4.2.

4.1 General Problem

Recently, the multiple testing problem has become an often discussed topic in the field of bio-sciences and is well-known in the field of statistics. For example, many trials in biomedical research generate large data sets. These are analyzed by performing many (sometimes thousands) of hypothesis tests. For example, see [Bender2001], [Dudoit2003], [Storey2001], and [Westfall1993]. In the field of computer science a well-known and well-discussed problem is the overfitting[4] problem. In general, overfitting occurs if a complex model is learned for only few data. But the problem can also occur if many possible models are tested and one model is chosen that fits the data well purely by chance (see [Domingos1999]); finally, this corresponds to the multiple testing problem.

The multiple testing problem arises when many hypotheses are tested. In common data mining algorithms (e.g. association rule mining and decision tree learning, see Sections 3.2 and 3.3) and other algorithms that perform an extensive search, the hypothesis tests are not necessarily expressed explicitly. The hypothesis tests are an implicit part of the search strategy. Nevertheless, this implicit part of the search strategy can be viewed as statistical tests.

For each single hypothesis test the error probability can be determined. However, often there is no direct control for the error probability when many (explicit or implicit) hypotheses are tested together. So it is necessary to use a statistical model that is able to control the error probability for all hypotheses.

> For example, in DNA micro-arrays thousands of tests are performed. A micro-array is a small biological slide with many spots. Each spot represents, for example, a specific gene. The micro-array experiments are aimed at finding differences between diseased and healthy tissues.

4 Overfitting is a problem that arises typically where the number of training examples is small or the learning process was performed too long. In these cases there is the danger that the learner adjusts to features of the training data, which have no (causal) relation to the target function and are only specific for the training data.

> For both, healthy and diseased tissue, micro-arrays are measured. Afterwards for each gene the spots on the arrays are tested for gene expression differences.

If a single hypothesis test at significance level α is performed (Section 2.2), the probability of the Type 1 error – rejecting the null hypothesis H_0, even though it is correct – is called **comparison-wise error rate** (CER) α. The probability of not rejecting the correct null hypothesis is $1 - \alpha$. Thus, for m independent tests, the probability of not rejecting all m null hypotheses when all are true is $(1 - \alpha)^m$. The **experiment-wise error rate** (EER) under the complete null hypothesis is the probability of rejecting at least one of the m independent null hypotheses when all are true. It is determined by EER $= \alpha_e = 1 - (1 - \alpha)^m$. This calculation only holds iff all hypothesis tests are independent[5]. If there is some correlation between the hypothesis tests the EER cannot be determined in such an easy way, because the EER depends on the hypothesis tests' structure of correlations. The EER is also called global level or family-wise error rate.

> As an example: 100 independent hypothesis tests are performed at significance level $\alpha = 0.05$. Thus, the EER will be EER $= (1 - (1 - 0.05)^{100}) = 0.9941$. So, one can almost be sure that, when testing 100 independent null hypothesis, at least one false significant result would be returned – even if the null hypothesis was true for all 100 tests.

The global null hypothesis that all individual hypotheses are true simultaneously is frequently of limited interest to the researcher. More interesting for a researcher is the probability of rejecting falsely at least one true null hypothesis, where it is not important which and how many other null hypotheses are true [Bender2001]. Such a probability is called **maximum experiment-wise error rate** (MEER) under any complete or partial null hypothesis. It means that the MEER is the probability of rejecting falsely at least one true single null hypothesis, where it is not important to know how many and which of the other null hypotheses are true. Multiple testing procedures that control the MEER also control the EER – but not vice versa. Thus, the best protection against wrong conclusions is provided by the MEER. The MEER is also called multiple level or family-wise error rate in a strong sense.

4.2 Common Solutions

This section reviews briefly some popular approaches for dealing with the multiple testing problem. These popular approaches use different concepts than our approach to deal with the multiple testing problem, which is introduced in Chapter 5, and they have some disadvantages, which could be solved by our approach. The Bonferroni correction is a quite simple statistical method which controls the MEER. A more recent approach is the false discovery rate, which uses a slightly different concept. Both approaches control the probability of failure for all hypotheses. Furthermore, a common approach in the field of computer science is given: cross validation. This approach allows not to control a certain error probability, but to estimate the probability of

5 Unfortunately, the assumption of independent hypothesis does not hold in practice. For the example given above it is quite unlikely that all expressions of genes are independent from the expressions of the other genes, since, for example, genes can overlap on the DNA helix. Thus, we get a strong dependence between the genes.

misclassification. The basic idea behind this approach is to avoid the overfitting problem, which is related to the multiple testing problem as mentioned in Section 4.1.

The following table, as in Section 2.2, displays the possible outcomes from m hypothesis tests.

Reality	Retain H_0	Reject H_0	Total
H_0 is true	U	V	m_0
H_0 is false	T	S	m_1
	W	R	m

The letters R through W represent the number of tests that fall into each category. For example, R is the number of significant tests, which reject the null hypothesis.

Bonferroni Correction

The **Bonferroni correction** is a very popular and quite simple approach to control the MEER – according to the table above, the MEER can be defined as MEER := P(V > 0). The basic idea of the Bonferroni correction is to correct the significance level α using m hypothesis tests, by dividing the significance level by the number of all hypotheses, $MEER \leq \alpha_{bon} = \alpha/m$. Equivalently, adjusted p-values could be calculated by $m \cdot p_i$, where p_i is the individual unadjusted p-value. This method is simple and it holds in all multiple testing situations, since it handles the worst-case where all hypotheses are independent. On the other hand it is very conservative (the number of false negatives becomes large) and it is not appropriate in situations where the number of tests is large, since the individual significance levels will become disproportional small (e.g. for significance level $\alpha = 0.05$ and 1000 hypothesis tests the adjusted significance level will be $\alpha_{bon} = 0.00005$). There are different improvements of the Bonferroni correction, for example a step-down procedure [Holm1979] and a step-up procedure [Hochberg1988], which are more powerful – but they are still conservative [Manly2004].

False Discovery Rate

The **false discovery rate** (FDR) is based on [Benjamini1995] and controls the expected proportion of Type 1 error among the rejected hypotheses. It offers a much less strict multiple testing criterion than the Bonferroni correction and leads to an increase of power. In [Storey2001] (this work describes also a method to estimate the FDR), the FDR is defined to be

$$FDR := E\left[\frac{V}{R}\middle| R>0\right] p(R>0) .$$

The FDR is used to determine so-called **q-values**[6]. There are different approaches to determine a q-value – a detailed discussion of procedures to control the FDR and determine the q-value can be found in [Dudoit2003]. Most FDR controlling procedures are designed for independent test statistics or they do not make use of dependence structures among the test statistics, but there are recently new approaches to control the FDR in a resampling-based way to deal with certain

6 The term q-value is inspired by the term p-value, since the FDR is slightly different than the CER or (M)EER. The q-value measures the significance level in some sense, too.

dependence structures, see for example [Dudoit2003]. The FDR provides a good alternative to MEER methods which is even less stringent of all corrections than the MEER methods and provides a good balance between discovery of statistically significant results and limitation of false positives. [Storey2001] gives also a proof that the FDR is suitable for so-called "loose dependence" – but it is not sure whether it will work with strongly dependent hypotheses.

Cross Validation

The concept of cross validation is a very common approach in the field of computer science. It works for supervised but not for unsupervised learning tasks. The cross validation estimates the error rate on new samples to provide a control for the overfitting problem. The data set is divided evenly into several partitions. A single partition will serve as the so-called test data set and the rest of the partitions will be used as the so-called training data set – this procedure is repeated until every partition was used once as test data set. The training data set is used to generate hypotheses and these hypotheses are evaluated with the test data set. The resulting error for each iteration is used to estimate the training error.

A more formal definition is given in [Wrobel2000]:

> Let E be a sample set. Let L be a learning algorithm. Then divide set E into m equal-sized subsets $E_1, \dots, E_m$. Generate the models (hypotheses) $h_1, \dots, h_m$ by: $h_i := L(E \setminus E_i)$
>
> Than the true error for L on E can be estimated by:
>
> $error(L,E) := \frac{\sum_{i=1,\dots,m} error(h_i)}{m}$, where error($h_i$) is some error measure for h_i on E_i.

Cross validation is only useful if the size of the data set is not too small (<10). There must be enough independent samples, so that it is not necessary to use all sample data at the same time to generate the hypotheses. Furthermore, the data mining method must be a predictive method.

Summary

There are different methods to control or estimate the quality of a result of a data mining algorithm: the Bonferroni correction, which controls the MEER by adjusting the significance level for each hypothesis test, the concept of the false discovery rate, which tries to estimate the proportion of falsely rejected null hypotheses under the rejected null hypotheses, and as third approach the cross validation, which estimates the probability of misclassification.

The Bonferroni correction is a very stringent method where the number of false positives is smaller than in the other approaches, but the number of results will also be very small. In general, the FDR seems to be a good solution – even if it is a different concept than the Bonferroni correction. There is also ongoing research to improve its behavior for strongly dependent hypotheses; but there is no approach that satisfies our interests of handling potentially very strongly dependent hypotheses. The cross validation approach is also not useful to solve our problem of strongly dependent hypotheses, since it provides no p-value and works only well if there are many independent samples. Furthermore, it is only applicable on predictive methods and not on descriptive like the the Apriori algorithm, which is given in Section 3.2.

Chapter 5 – Framework Definition

In Section 5.1 our approach to assess the multiple testing error of a data mining approach is motivated and the basic idea is introduced. The basic conditions and definitions are given in Section 5.2. In Section 5.3 the basic algorithm is presented and is described in detail.

5.1 Motivation

Since in bio-scientific research the number of independent samples is often very small (≤ 15) and the effort to obtain more samples can be expensive or ethical questionable, there is the need for a method which can deal with only a few independent samples. Furthermore, (several) thousands of hypotheses are tested to generate a result or result set, which includes many (in some cases strongly) dependent hypotheses. Our approach provides a measurement how useful the complete hypothesis set, which was generated by some data mining approach, can be. This method provides no information about the significance level for each single hypothesis. The evaluation of the single hypotheses is seen as further step in the research process, which should be performed only if our method says: "it is likely that the hypothesis set contains useful hypotheses".

The main goal of our approach – to determine a certain error probability of the complete result of a data mining algorithm – is quite similar to the methods which are given in Section 4.2, but we go a different way. We suppose that a set of hypotheses can also be generated by a data mining process from sample data, which does not contain useful information. To determine whether a result is useful or not, we use a permutation test (Section 2.3) on the data. The permutation step generates a distribution under the assumption that the data mining algorithm produces a noisy result (for the given data set). This distribution is used for standard hypothesis testing (Section 2.2).

The basic idea of our approach is to use standard data mining algorithms without any change to them and to evaluate their results in an appropriate way. The evaluation should provide a statistical measure which is comparable to the standard p-value. Due to this we use a resampling approach, which is adopted from [Westfall1993]. We generate a permutation distribution over the data set to perform (with the obtained distribution) classical statistical hypothesis testing by applying the chosen data mining algorithm to each permutation of the original data set again. Since the results of the data mining algorithm are usually not suitable as variables to derive a distribution, there is the need for a function which maps the results to the space of real numbers. From

hereon we call such a function an evaluation measure. The evaluation measure depends on the data mining algorithm. There are two different groups of evaluation measures: those which are increasing if the data contains more information and those which are decreasing if the data contains more information (a detailed definition is given in Section 5.2).

5.2 Basic Conditions

Since our approach is indeed not applicable to every problem or data set, respectively, and to every data mining method, there is the need to specify basic conditions where our approach is supposed to work well. Due to the need to perform a permutation test (the basic concept of permutation tests is given in Section 2.3) with the original data set, it is necessary that the data sets consists of at least two different classes[7]. These different classes should have more or less the same proportions in the data. Since the resampling distribution, which is generated by the permutation test, is in general not symmetric, a decision is necessary, which kind of hypothesis test (Section 2.2) – lower one-sided testing or upper one-sided testing – should be performed.

On the algorithmic side, our approach is only applicable to data mining algorithms, which can deal with different class labels. These algorithms should produce a symbolic output – for example, association rule mining or decision tree learning as given in Section 3.2 and Section 3.3 – since this symbolic output facilitates the specification of the so-called evaluation measure (see below). The definition of a suitable evaluation measure for the results of the data mining algorithm becomes much easier if the data mining algorithm returns symbolic output, since such output helps to interpret the results in a better way. In general the definition of a evaluation measure becomes a crucial aspect in our approach and it is important that the result of the evaluation measure is intuitive. Furthermore, it is important that there is no effect of learner parameters to the evaluation measure (e.g. limiting the tree size for decision tree learner and using the tree depth as evaluation measure).

Evaluation Measure

Due to the fact that most types of results of data mining algorithms cannot be used directly for statistical testing, there is the need to map these results to a space in which statistical inference becomes much easier. Such a mapping is provided by the so-called evaluation measure. The evaluation measure em is a function which maps the result of a data mining algorithm for a certain sample data set to a single real value. The basic idea behind such a function is to provide a measurement (of quality or quantity) of the results of a data mining process. This measurement must provide possible information about the strength of information in a given data set. The selection of a suitable evaluation measure becomes a crucial aspect for our experiments as one will see in Section 6.3.

Formally, the data mining approach in combination with the evaluation measure can be seen as (quite complex) test statistic – at least if the data mining approach is deterministic. Both together

7 It is obvious that a permutation over a vector of class labels, where only one class label exists, does not make sense.

compute a function of the sample values. The distribution of this statistic is estimated with a resampling procedure – the permutation test (see Section 2.3). This test determines whether there is a dependence of the target attribute on the descriptive attributes and this dependence can be captured by the data mining approach.

The following gives a more formal definition of the evaluation measure:

> Let Σ be a given data set and let the data mining algorithm be a function $f : \Sigma \rightarrow \Gamma$, where Γ is the learner's output. Then the evaluation measure is a function $em : \Gamma \rightarrow \mathbb{R}$.

An important property of the evaluation measure is its so-called orientation. Intuitively, there are two possibilities for the value of the evaluation measure: it can increase if the strength of a information pattern increases in a data set – or the other possibility: it can decrease if the strength of a information pattern increases in a data set. The first kind will be called positively oriented and the second will be called negatively oriented, respectively. Determining the orientation is important for the decision whether a lower one-sided test or a upper one-sided test should be performed (Section 2.2).

The following gives a more formal definition of the orientation of an evaluation measure:

> Let $X = \{X_1, \ldots, X_n\}$ be a set of descriptive attributes and each attribute X_k can take several values x_{kl}. Furthermore, let Y be the target attribute with at least two different attribute values. Let Ω^0 be some population without any dependencies between the descriptive attributes X and the target attribute Y and let Ω^* be some population with some dependencies between the descriptive attributes X and the target attribute Y. For both, Ω^0 and Ω^*, the descriptive attributes X must have the same fixed distribution.
>
> Then, by drawing samples Σ^0_i from Ω^0 and Σ^*_j from Ω^* and applying some data mining method f on these samples, one can estimate the expected values of the evaluation measure em for the samples of both populations:
>
> $$E[em(f(\Sigma^0_i))] = \theta^0 \text{ and } E[em(f(\Sigma^*_j))] = \theta^*$$
>
> An evaluation measure em is called efficient, iff $\theta^0 \neq \theta^*$ holds. Furthermore, em is called negatively oriented, iff $\theta^0 > \theta^*$ holds, and em is called positively oriented, iff $\theta^0 < \theta^*$ holds, respectively.

Calculation of p-Value

One of the main goals of our approach is to provide a suitable measurement of the error probability for a result of a data mining algorithm for a given data set. This measurement should be transparent for biologists or other scientists. The p-value according to the null hypothesis that the results of the data mining algorithm represent only random data is estimated. Such a p-value provides such a suitable measurement for the result of our approach. For the estimation the permutation distribution is generated (see Section 2.3 for theoretical details and Section 5.3 for our realization).

The computation of the p-value $p_{est.}$ depends on the orientation of the evaluation measure. For a data set Σ and its evaluation measure's value em_Σ it is to determine in how many cases the

resampling distribution of the evaluation measures $em_{\Sigma[i]}$ for the permuted data sets Σ_i, where $i = 1, \ldots, n$, exceeds em_{Σ}.

Formally, the computation is performed by:

$$p_{est.} = \frac{\left|\{em_{\Sigma[i]} | em_{\Sigma[i]} \geq em_{\Sigma},\ i=1,\ldots,n\}\right|}{n}$$, for positively oriented evaluation measures.

$$p_{est.} = \frac{\left|\{em_{\Sigma[i]} | em_{\Sigma[i]} \leq em_{\Sigma},\ i=1,\ldots,n\}\right|}{n}$$, for negatively oriented evaluation measures.

5.3 Basic Algorithm

This section combines all theoretical basics and definitions given in the preceding sections. According to the procedure to derive a resampling distribution (which is given in Section 2.3) and the notation which is invented in Section 5.2, the basic algorithm can be formulated as follows.

Our algorithm consists of three basic steps:

(1) The data mining algorithm is applied to the original data set and for its result the evaluation measure is computed. As an optional processing step to compute the evaluation measure it could be necessary to filter the result. For example, one may have to reduce the result set of the Apriori algorithm to rules which contain the target attribute.

(2) For n iterations, a permutation of the target attribute of the original data set is generated and the data mining algorithm is applied on the resulting new data set. As in step (1) the evaluation measure is computed for the result of the data mining algorithm, which could also be filtered.

(3) In the last step, the p-value $p_{est.}$ for the whole experiment is computed according to the formula, which is given in Section 5.2.

The following gives a more pseudo-code-like description of our approach:

```
run learner with original data set Σ to derive result Γ_Σ
(optional step)[8] filter result Γ_Σ
calculate evaluation measure em_Σ
for i = 1, ... , n
    permutate target attribute of original data set randomly
    run learner with permuted data set Σ_i to derive result Γ_Σ[i]
    (optional step) filter result Γ_Σ[i]
    calculate evaluation measure em_Σ[i]
calculate p-value p_est.
```

8 The optional step of filtering the results of the data mining algorithm is only necessary for data mining algorithms like the Apriori algorithm to reduce the result set to the interesting results which are related to the target attribute.

Chapter 6 – Experimental Results

In Section 6.1 of this chapter a general experiment description is given – this includes the technical parts of our implementation and a short description of possible results of our approach. Section 6.2 presents the data sets which are used for our experiments and in Section 6.3 our experiments and their results are discussed. A summary of all results of this chapter is given in Section 6.4.

6.1 Technical Details

Detailed Experiment Description

Our algorithm to assess the error probability of the result of a data mining approach for a given data set is applied to different data sets and different data mining algorithms are used. The basic algorithm is introduced in Section 5.3. The data mining algorithms are given in Chapter 3 and a detailed description of the data sets that are used is given in Section 6.2. All programs for analyzing the different data sets are implemented in Perl5 (see Appendix A for an example). As learning algorithms the free implementations of the Apriori algorithm and decision tree learner of Christian Borgelt are used. These implementations can be obtained from [Borgelt2005]. The algorithmic details for both data mining approaches are given in Section 3.2 and Section 3.3.

The whole procedure can be seen in Figure 5. First, the input data of the learning algorithm is preprocessed by a converter which changes the format of the UCI-ML data sets (for details see Section 6.2). The preprocessing is necessary, since the learner algorithms need a slightly different input format than the default format in the UCI-ML archive. The converter, which is implemented in Java, reads a configuration file (an example is given in Appendix A) which allows to replace continuous attribute values by discrete values and to omit certain attributes. Furthermore, the converter generates additionally the permuted data sets which are needed to determine the resampling distribution. The number of permuted data sets is set to 200 for all experiments.

In the second step, the unpermuted converted data set is used as input for the learner algorithm. Optionally, the output of the learner algorithm is filtered. This sub-step is necessary for the association rule mining in order to consider only rules containing the target attribute – all other rules are uninteresting to solve the problem and should be omitted. For the (filtered) output the evaluation measure is computed. Finally, the resulting value will be used to compute the p-value.

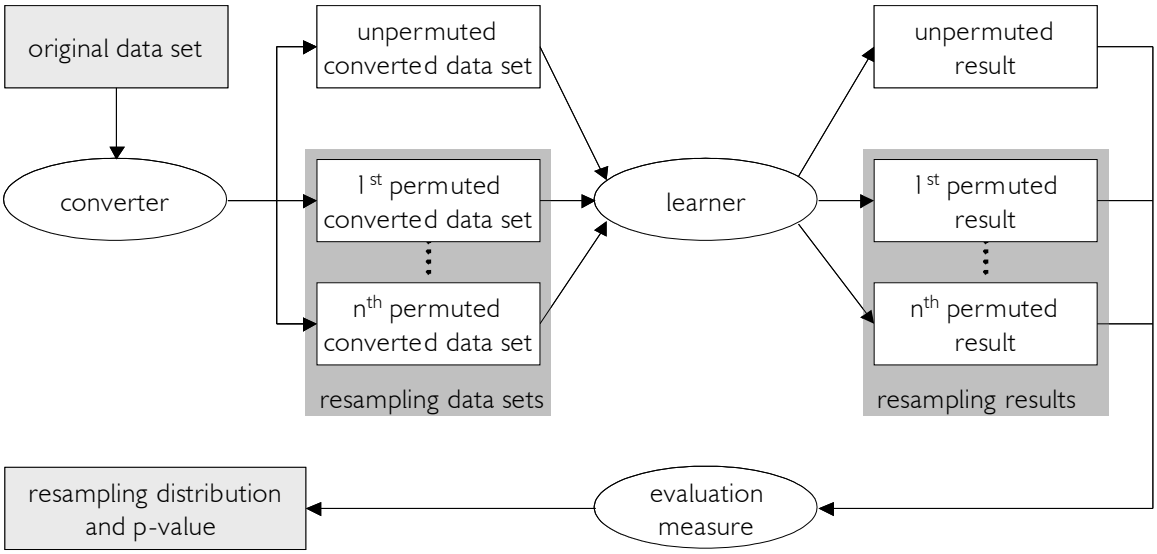


Figure 5: Realization of the resampling procedure.

The third step is the resampling procedure. For each permuted converted data set from the first step the output of the learner algorithm is determined. Each result can be filtered and then the evaluation measure is determined. All evaluation measure values for the permuted data sets are stored in order to generate a histogram and to compute the p-value for this experiment.

Apriori Algorithm

The Apriori implementation from [Borgelt2005] is used for our experiments (details for the Apriori algorithm are given in Section 3.2). This implementation allows us to use different output types (item sets, closed item sets, maximal item sets, association rules, and association hyperedges). Most of our experiments are performed with two possible output types – frequent item sets and association hyperedges. Without loss of generality, we will call all output types from hereon rules. For both output types two evaluation measures are used – the number of rules with the target attribute and the number of non-ambiguous rules. The first evaluation measure is simply the number of rules which are containing the target attribute. The second evaluation measure only counts the rules which are containing the target attribute and which are unique in the rule set if one omits the target attribute. Both evaluation measures are chosen, since the number of rules that are found by the Apriori algorithm increases, if the dependencies between the descriptive attributes and the target attribute become larger. Due to this, these evaluation measures are positively oriented. As a further parameter, for the Apriori implementation, the result set consists only of rules with at least three items and not more than seven items. For all other possible parameters the default values are used. Thus, the minimum support is ten percent.

Decision Tree Learner

As a second data mining algorithm the decision tree learner implementation from [Borgelt2005] is used for our experiments (Section 3.3 gives algorithmic details about decision tree learning algorithms). In this case, the experiments are only performed with the census data set and the mushroom data set – the results are similar to the results using the Apriori algorithm, so we did no further experiments. For both data sets the experiments are performed with the default

attribute selection measure (the information gain ratio – for more details see [Borgelt1998]) and with the information gain as attribute selection measure. Since the structure of the decision tree learner results is different to the result structure of the association rule learner, there is the need for another evaluation measures. For our experiments with the decision tree learner, the tree depth and the number of nodes are used as evaluation measures. Both values are included in the output of the implementation – so there is no need for further computation. These values are chosen, since the number of nodes or the tree depth, respectively, increases, if the dependencies between the descriptive attributes and the target attribute become weak. The less dependencies are in the data, the more splits (inner nodes) are necessary to describe all the data. This holds as long as the decision tree is unpruned – as in our experiments[9]. Due to this, these evaluation measures are negatively oriented. All other default parameters of the implementation are retained.

Expected Results

Figure 6 shows the possible outcomes for our experiments, where the hypothesis test is an upper one-sided test. These outcomes represent the result of a hypothesis test (see Section 2.2) with the null hypothesis that there are no dependencies between the descriptive attributes and the target attribute in the data set and the alternative hypothesis that there are some dependencies between the descriptive attributes and the target attribute in the data set. The p-values are computed as explained in Section 5.2.

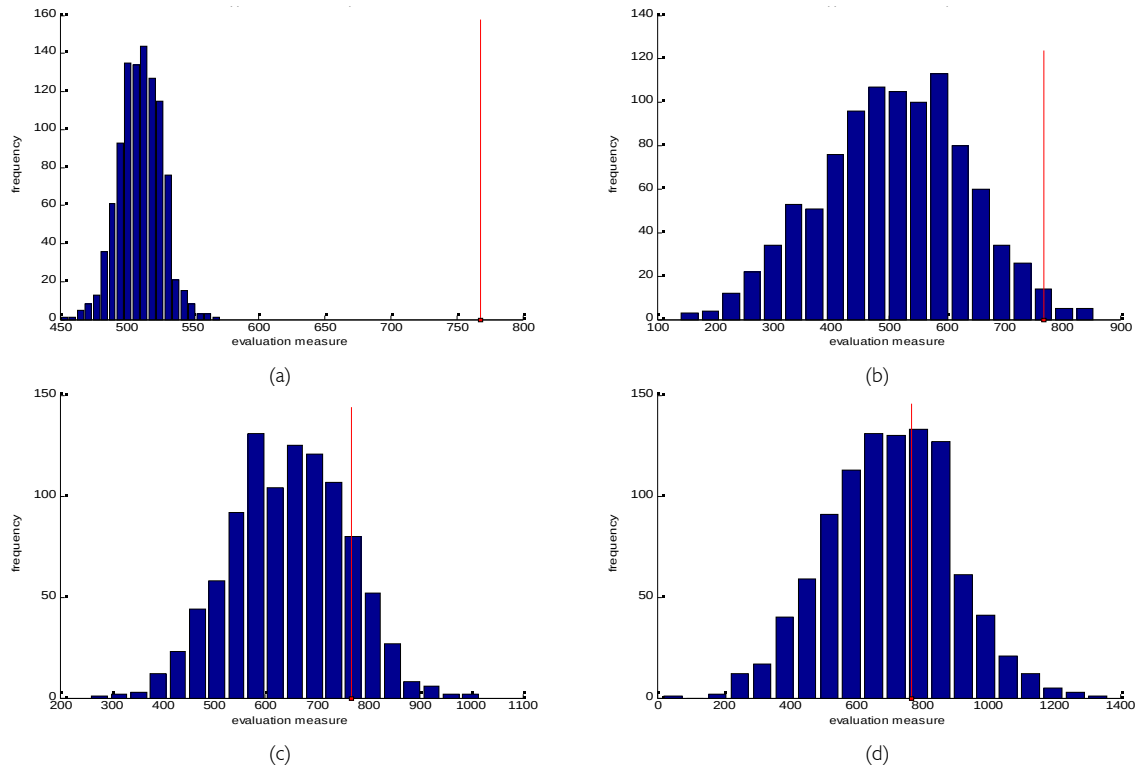


Figure 6: Expected results for the resampling procedure: (a) is an ideal result, (b) provides good evidence to reject the null hypothesis, (c) is an ambiguous case, and (d) provides no evidence to reject the null hypothesis. The red line represents the value of the original data set and the blue bars represent the permuted data sets as a histogram of the resampling distribution.

9 It is possible that this behavior of the evaluation measures will change, if one uses pruned decision trees instead. This could be seen as a part of further experiments.

Figure 6.a shows an ideal case where the resampling distribution has a small variance and the distance between the resampling distribution and the measured value (this value is represented by the red line) is quite large. A good result, as in Figure 6.b, would be that the number of elements of the resampling distribution, which are larger than or equal to the measured value, is quite small. In such a case we would also expect that the variance of the resampling distribution is greater than for Figure 6.a. Figure 6.c shows an ambiguous case – in such a situation there is the need of further discussions with the experts, if they really believe their results are containing interesting information or not. Figure 6.d shows a poor case where it is very unlikely that the null hypothesis is false.

6.2 Data Set Descriptions

Two different types of data sets are used to analyze the applicability of our approach. The first group of data sets are well-known data sets like the census data set and the mushroom data set. These data sets demonstrate that our approach can deal with "real world" examples. The second group of data sets are (semi-)synthetically generated data sets; the MyMONKS data set and the CensusHS-Induced data set. The CensusHS-Induced data set is used to demonstrate the behavior of our approach if there is no difference between the different target attribute groups. The MyMONKS data set serves as an example to show the influence of the strength of dependencies between the single attributes of a data set on our approach.

Census Data Set

The census data set[10] is derived from the 1990's census data from the US Census Bureau and it is published in [UCI-ML1998]. This data set consists of 48842 instances (the training data and the test data are combined). One instance consists of 15 continuous or discrete attributes. The continuous attributes are discretized (for details see the configuration file in Appendix A) and the attributes "fnlwgt" (unknown meaning) and "education-num" (this attribute is equivalent to the attribute "education") are omitted. The original prediction task for this data set is to determine whether a person makes over 50,000 $ a year – but we used the prediction of the educational level instead. The prediction task is changed to demonstrate that our approach can deal not only with two but also with multiple class labels.

Mushroom Data Set

The mushroom data set is also published in [UCI-ML1998]. This data set includes descriptions of hypothetical samples corresponding to 23 species of mushrooms. Each species is identified as "definitely edible", "definitely poisonous", or of "unknown edibility" and "not recommended". The mushrooms with unknown edibility and those which are not recommended are combined with the poisonous class. This data set consists of 8124 instances where each instance is represented by 22 discrete attributes. The prediction task is to determine whether a mushroom is poisonous.

10 The census data set is newly renamed into adult data set.

CensusHS-Induced Data Set

The CensusHS-Induced data set is based on the census data set. The data set consists of 15784 instances from the census data set. All instances of the census data set which do not belong to the class "HS-grad" are omitted. The resulting data set is divided randomly into two balanced artificial groups "HS-grad0" and "HS-grad1". For further experiments special attribute values are 'induced' into the CensusHS-Induced data set with different intensities. For each of both groups, the values of certain attributes are overwritten by other fixed values with a certain probability (see Appendix A for more details). We used the prediction of the educational level – as for the census data set.

MyMONKS Data Set

The MyMONKS data set is based on the first of the MONK problems which are also published in [UCI-ML1998]. The data set consists of eight different attributes; one of them is a unique ID and the class attribute is determined as a function of three of the remaining six attributes (for details see Figure 7). As a training data set 50 percent of the data set are chosen randomly.

To show the influence of the strength of dependencies among the single attributes on the resampling distribution three different types of induced relationships are used: (1) a_5 is a copy of a_4 with a probability p, (2) a_4 is a copy of a_3 with a probability p, and (3) a_5 is a copy of a_4 with a probability p_1 and a_4 is a copy of a_3 with another probability p_2. (1) is called type-1 structure, (2) is called type-2 structure, and (3) is called type-3 structure, from hereon.

attribute values:
$class = \{0, 1\}$, $a_1 = \{1, 2, 3\}$, $a_2 = \{1, 2, 3\}$, $a_3 = \{1, 2\}$, $a_4 = \{1, 2, 3\}$, $a_5 = \{1, 2, 3, 4\}$, $a_6 = \{1, 2.\}$, Id = unique symbol
class attribute function:
if ($a_1 = a_2$) or ($a_5 = 1$) then
class := 1 else class := 0

Figure 7: Basic definition of MyMONKS data set.

As one can see in Figure 7 the class label is determined by only three of the six attributes. The other attributes represent some kind of noise. In Section 6.3 the influence of such noise attributes[11] is shown.

6.3 Data Set Results

As mentioned in Section 6.2 we used different types of data sets: synthetically generated data sets, original data sets from the UCI Machine Learning Repository ([UCI-ML1998]), and a data set which is currently investigated by MelTec (this data set is discussed in Section 7.4). The experiments with the synthetically generated data sets are only performed with the association rule mining algorithm as a data mining approach, since the experiments are computationally expensive and quite time consuming. The experiments for the "real world" data sets are performed with both data mining approaches, association rules mining and decision tree learner, which are presented in Section 3.2 and Section 3.3.

11 We will call attributes which does not influence the class label from hereon noise or noise attribute.

Synthetical Data Sets

As already mentioned above, we used two synthetically generated data sets. The first one – the **CensusHS-Induced data set** – is based on a "real world" data set (the census data set), but we modified it as described in Section 6.2. It is used to demonstrate the behavior of our approach for data sets with patterns of data that are expressed with different strength in the data sets. The strength of a pattern is determined by the probability of 'inducing' certain other attribute values.

Figure 8 and Figure 9 show, as examples, box plots for the p-values for induced patterns that are expressed with different strength in the CensusHS-Induced data set. As lowest strength of a pattern 10% is used, since the Apriori algorithm uses by default 10% as minimum support. For values larger than 15% the p-values are small in all cases. For each of the six box plots per line 20 data sets are generated with the specified strength of the pattern. For each of these 120 data sets 200 permuted data sets are generated and the p-value is computed. For every data set the result of the Apriori algorithm is determined for search depths between three and seven[12].

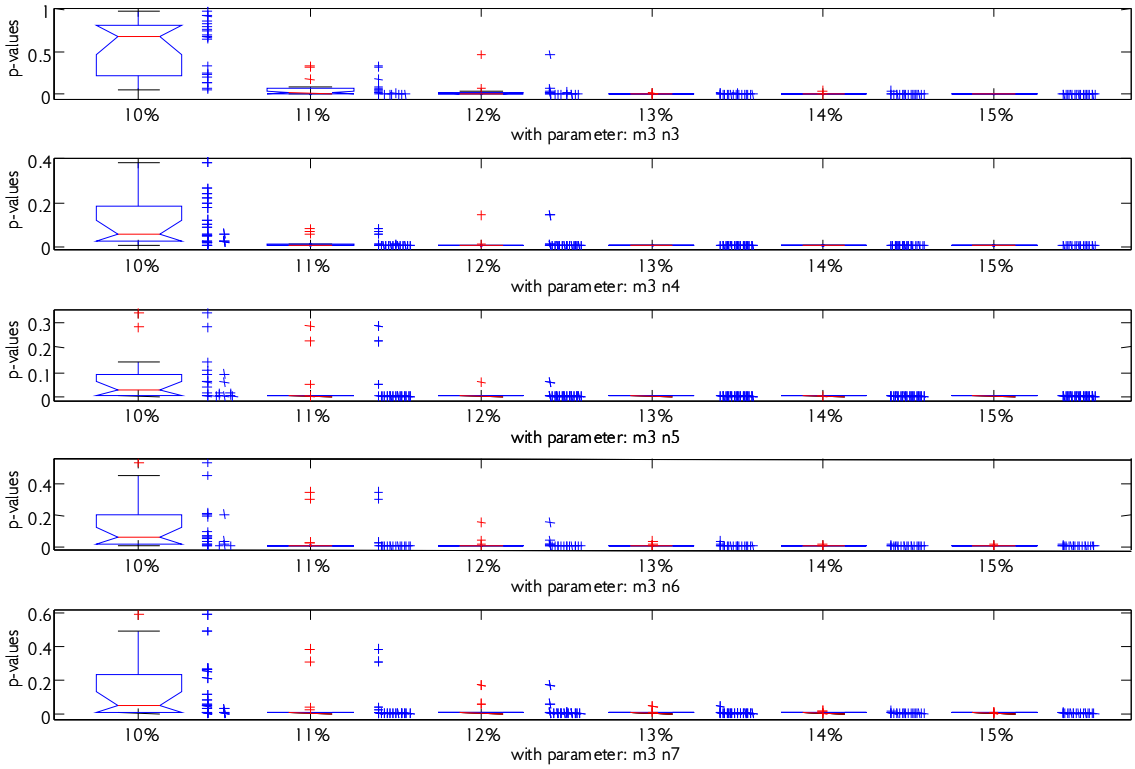


Figure 8: Example for box plots of the p-value for differently strong expressed patterns (the strength of a pattern for this plot is between 10% and 15%) in the CensusHS-Induced data set and different search depths (from search depth 3 to up to search depth 7) of the Apriori algorithm. As evaluation measure the number of non-ambiguous rules with target attribute is used.

As one can see in Figure 8, where the number of non-ambiguous rules with the target attribute is used as the evaluation measure, for a strength of the pattern of 10% the range of the p-value is between zero and one for all search depths – but for larger search depths the range will become

12 The Apriori implementation of Christian Borgelt accepts the parameter "m" to determine the minimal number of items in the result set and the parameter "n" to determine the maximal number of items in the result set. So, for example, "n3 m4" means "the result set consists of rules with at least three items and at most four items".

smaller and closer to zero. This is the expected result, since the strength of pattern of 10% and the minimal support, which is used in our experiments, are equivalent, so the chance to get a good result is small. In general, for an increasing strength of the induced patterns the variance of the results for the p-values will become smaller and the p-values will get closer to zero.

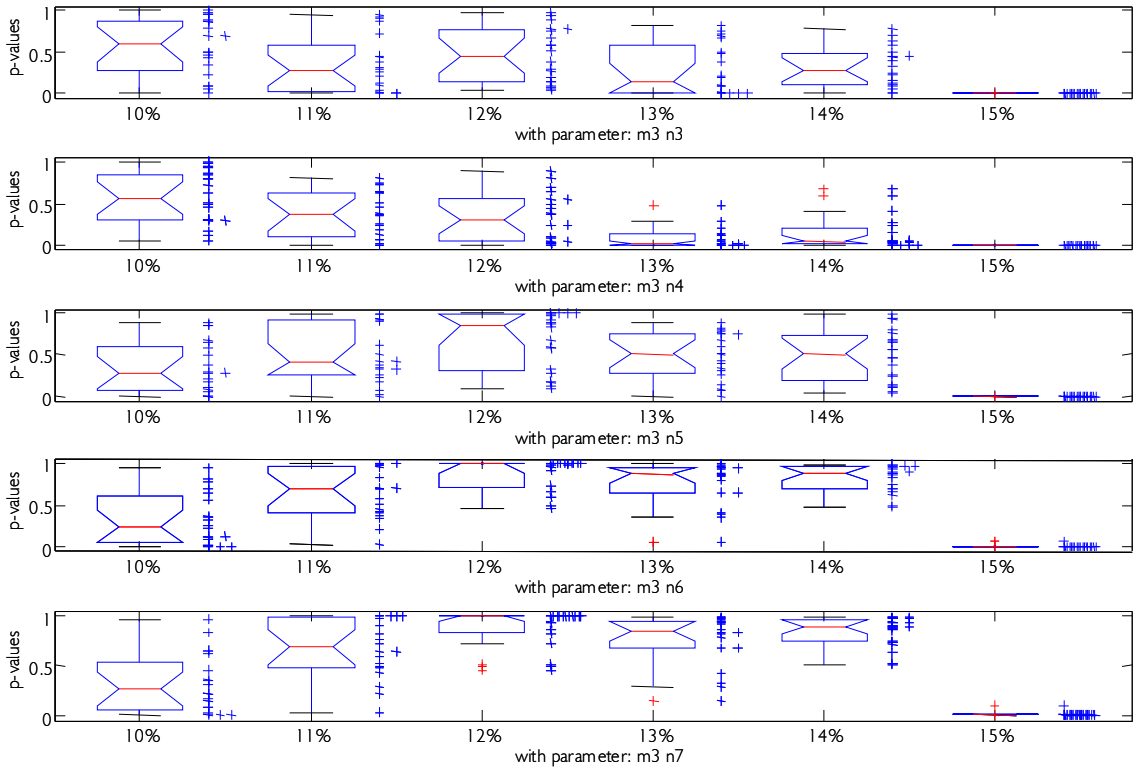


Figure 9: Box plots for the same data sets as in Figure 8, but with the number of all rules with the target attribute as evaluation measure.

Figure 9 serves as an example for a poor evaluation measure for the CensusHS-Induced data set. The strange behavior of the p-values occurs, since the permuted data sets consist of many ambiguous rules, where an ambiguous rule is a rule, which differs only in the target attribute value. As an example there is a rule x_1, x_2, y_1 and another rule x_1, x_2, y_2, where x_1, x_2 are some attribute values and y_1, y_2 are two different class labels. So, the number of rules for the permuted data sets will become larger than for the original one and the p-value will get closer to one than to zero. Another strange effect is that for a strength of pattern larger than 14% the p-values suddenly drop to values so as one would expect. We assume that this effect occurs because of a lack of influence of the ambiguous rules if the strength of the patterns will become larger than 14% - at least this holds for our experiment with the CensusHS-Induced data set.

As a second interesting part, the effect of dependencies among the attributes in a data set was investigated. To estimate the behavior of our approach for such data, the **MyMONKS data set** (see Section 6.2) is used. Figure 10 gives example plots for the MyMONKS data set with the number of non-ambiguous rules as the evaluation measure. These plots are generated for the three different types of the MyMONKS data set. Type-1 represents a dependency between the attributes a_4 and a_5 – this is equivalent to a direct influence of the noise (see Section 6.2) on the class label; type-2 represents a dependency between the attributes a_3 and a_4 – an intra noise attributes dependency, and type-3 is a combination of both, type-1 and type-2, dependencies.

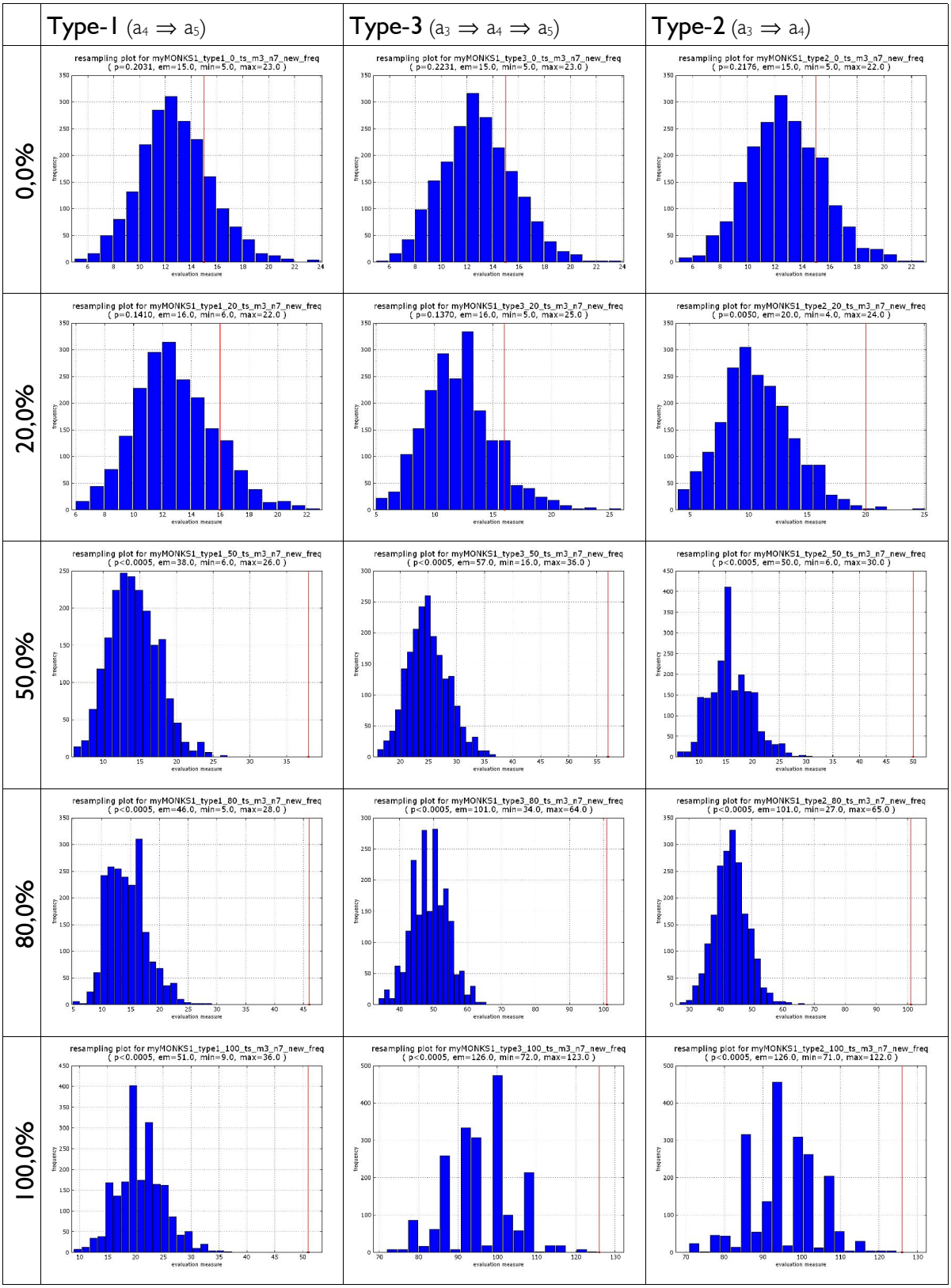
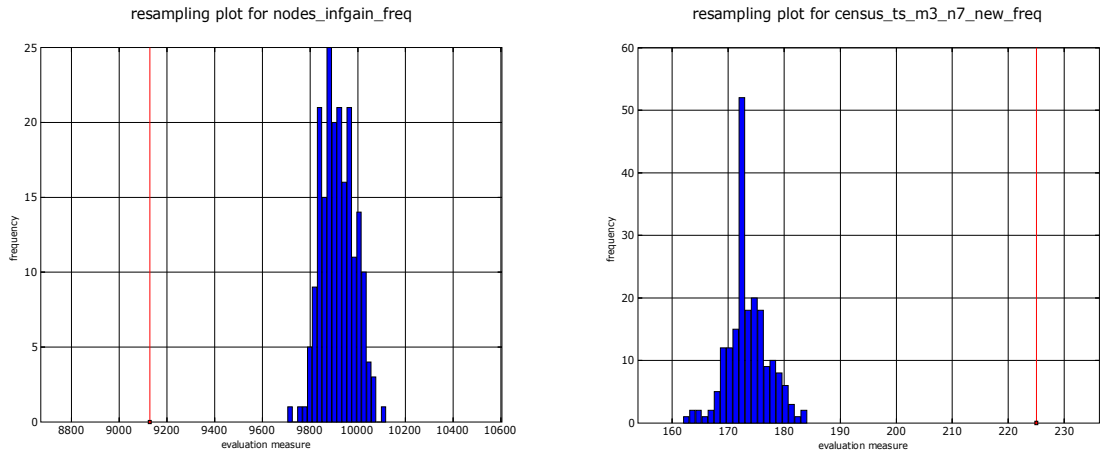


Figure 10: Plots for the three types of the MyMONKS data set. The percentage represents the probability that a_j is set to the same value as a_i, according to the type definition.

In general one can say: the greater the dependency among the attributes in the data set the larger the gap will be between the resampling distribution and the value of the original data. This holds especially for the type-1 dependency structure. For the type-2 and type-3 structures and for very strong dependencies among the attributes the gap will become smaller and the resampling distribution becomes more jagged – at least this holds for the Apriori algorithm as a data mining approach. Another general effect is that the larger the dependency the larger the variance of the resampling distribution will be – especially for the data sets with type-2 and type-3 structure.

UCI-ML Data Sets

The census data set and the mushroom data set can serve as typical results for our approach. These data sets are used to show the general suitability of our approach for "real world data". Both data sets are explained in Section 6.2. For both data sets the results are quite similar: there is a large gap between the original value and the resampling distribution.



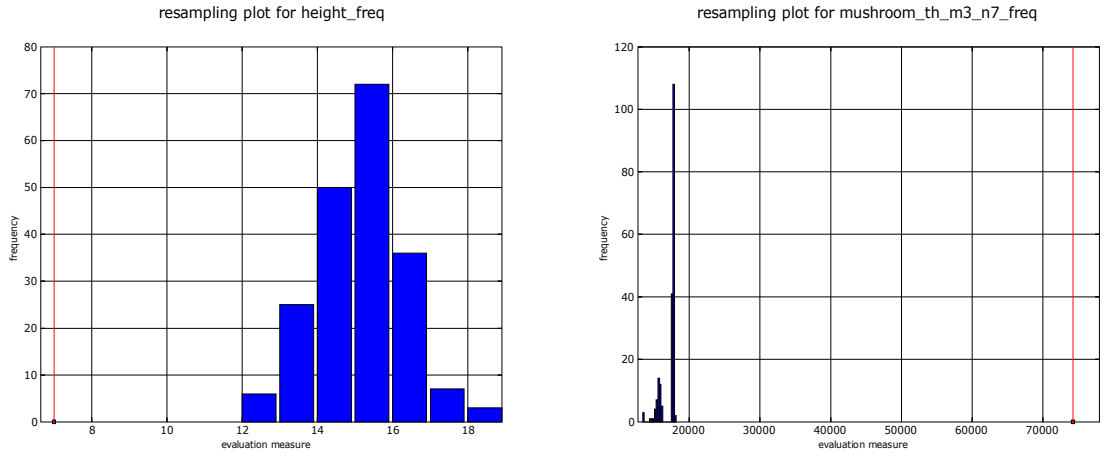
(a) (b)

Figure 11: Resampling plots for the census data set. (a) is a plot for a decision tree experiment, where the number of nodes is used as evaluation measure and the decision tree is generated using the information gain measure. (b) is a plot for an association rules experiment, where the number of non-ambiguous rules is used as the evaluation measure and Apriori's results consist of item sets.

In Figure 11 two different plots for the **census data set** are shown. Figure 11.a shows the plot for the number of nodes as the evaluation measure, where the decision tree is generated using the information gain measure. The same experiments are performed with the standard measure for the decision tree learner implementation, which results in quite the same plot. The tree depth in the case of the census data set is not a suitable evaluation measure, since the depth of the tree will be around twelve for both, the original data and the permuted data. Figure 11.b shows the plot for the number of non-ambiguous rules as evaluation measure, where the result of the Apriori algorithm consists of frequent item sets. As for the other figure the three other combinations of evaluation measure and learner parameter result in quite similar plots. For both figures the distribution looks normally distributed and the gap between the resampling distribution and the original data value is quite large – so we have a favorable result.

Figure 12 gives two example plots for the **mushroom data set**. Figure 12.a shows the plot for a decision tree experiment, where the tree depth is used as evaluation measure and the standard parameters are used to induce the decision tree. This is one of the rare situations in which the

tree depth is suitable as an evaluation measure. As one can see, the suitability of an evaluation measure depends not only on the data mining algorithm but also on the data set structure. Figure 12.b shows the plot for the Apriori algorithm, where the result consists of hyperedges, and the simple number of rules with the target attribute is used as evaluation measure. In this plot one can see the effect of strong dependencies among the attributes as already shown for the MyMONKS data set. For the other possible combinations of the learner's parameters and evaluation measures the results are, as for the census data set, quite the same – only the number of nodes as evaluation measure (as shown in Figure 13) shows a much larger gap between the original data value and the resampling distribution. This is because the number of nodes increases exponentially with the tree depth (at least for roughly balanced trees).



(a) (b)

Figure 12: Resampling plots for the mushroom data set. (a) is a plot for a decision tree experiment, where the tree depth is used as evaluation measure and the decision tree is generated using the default parameters. (b) is a plot for an association rules experiment, where the simple number of rules is used as evaluation measure and Apriori's results consist of hyperedges.

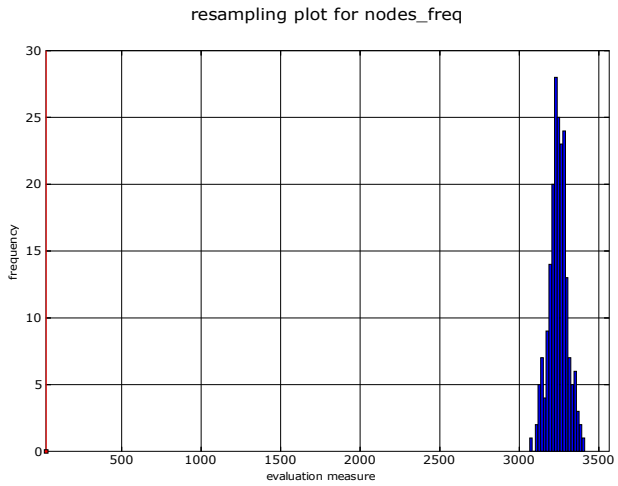


Figure 13: Plot for mushroom data set with the number of nodes as evaluation measure.

All plots and the resulting p-values for our experiments with the UCI-ML data sets (the census data set and the mushroom data set) can also be found in Appendix B.

6.4 Discussion and Summary

The experiments have shown that our approach provides suitable information about the quality of the result of a data mining algorithm. The suitability is shown for real world data sets as well as for synthetically generated data sets. For both one can see, that the gap between the value for the original data set and the permutation distribution becomes large if there is more information included in a data set – this means the strength of a pattern increases.

During our experiments it turned out that the selection of a suitable evaluation measure is a crucial aspect. For example using the Apriori algorithm it can occur in some surprising cases that for the results, which are found by the data mining algorithm, the resampling distribution consists of more rules than the original data set, but most of the rules of the resampling distribution are ambiguous. That is, there are rules which describe different classes with the same properties and property values. Surprisingly, we discovered this effect while we examined the results for the semi-synthetically generated CensusHS-Induced data set – so this effect might be an artifact of the data generation process that will not occur in real world examples. Indeed, both evaluation measures for the Apriori algorithm seem to be suitable for all other data sets, which are tested during our experiments. There exists also an effect of certain learner parameters on the evaluation measure – so it is obvious that limiting the tree depth for a decision tree has an unfavorable influence on the tree depth as the evaluation measure. There might be some similar parameter combinations which are not so obvious. But in general and especially for "real world" data sets our approach provides the expected results.

The following table summarizes the data mining algorithm and possible evaluation measures:

learner	evaluation measure	orientation	remarks
Apriori (frequent item sets)	number of rules with target attribute	positive	in general a good measure, but ambiguous rules (rules which differ only in target attribute) can impair quality of p-value
	number of non-ambiguous rules with target attribute	positive	good measure with no known disadvantages w.r.t. the calculation of the p-value, but the result size may be shrunk
DTree (decision trees)	tree depth	negative	critical measure – in most cases it does not distinguish the results, since the tree depth will be equal for both, resampling and original data; additionally, if the depth is bounded by learner parameter this measure is not useful; strong dependence on data set
	number of nodes in tree	negative	good measure with no known disadvantages w.r.t. the calculation of the p-value

Chapter 7 – Biological Application

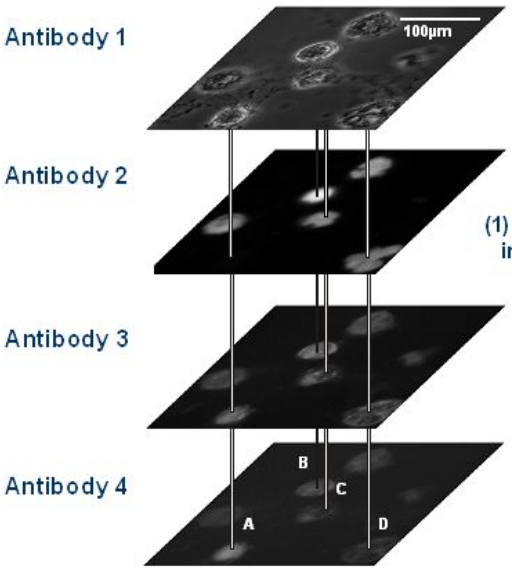
In this chapter an example for the application of our approach to biological data is given. Section 7.1 introduces the basic technology which is used to process the biological samples. A short introduction into MelTec's data analysis methods is given in Section 7.2. The description of our resampling experiment with the MotifFinder technology is given in Section 7.3 and in Section 7.4 the result for a certain MelTec data set is discussed.

7.1 The MELK Data

MelTec's proprietary MELK[13] robotic technology produces complex protein data in the field of bioscience. Most applications using MELK-data are such that two biological samples are compared. Determining the location and the kind of differences in the two samples may indicate the difference in biological functioning. Commonly, untreated samples are compared with samples whitch are treated with a certain drug, or healthy tissue is compared with diseased tissue. The difference between these samples indicates the effect of the drug, or the cause of the disease.

The process to generate motifs from images is shown in Figure 14. The MELK data is based on a biological sample for which a stack of n fluorescence images is produced. Each of the n images corresponds to one biological marker (antibody) standing for a certain protein, a protein class or a biological structure (e.g. cell nucleus). The images within the stack are perfectly aligned – so the pixels with the same coordinates correspond to the same biological region. (1) The aligned pixels are collected. Then, for each image a binary image is derived by setting a threshold (2), and for each stack pixel, a binary vector is generated, which represents the appearance of all markers for the same biological region. Then these binary vectors are collected into so-called Combinatorial Protein Patterns (CCPs) and the CPP's frequencies are calculated (3). When looking for differences between two populations, not all n markers might be involved. In other words: the difference can be characterized already by a subset of the n markers. Thus, in step (4) the concept of a motif is introduced. A motif allows wildcards (*) in the description of a certain state of co-localization. A wildcard is equivalent to the statement that the corresponding marker does not contribute to the difference. As for CCPs the motif's frequencies are determined as well.

13 MELK is an acronym, which stands for "Multi Epitop Ligand Kartierung".



(1) Collecting photonic intensities for each pixel

Gray-Value Table	Pixel A	Pixel B	Pixel C	Pixel D
Antibody 1	50	55	130	80
Antibody 2	10	400	300	280
Antibody 3	200	145	160	180
Antibody 4	210	240	90	100

(2) Setting a threshold (150)

Binary Table	Pixel A	Pixel B	Pixel C	Pixel D
Antibody 1	0	0	0	0
Antibody 2	0	1	1	1
Antibody 3	1	0	1	1
Antibody 4	1	1	0	0

(3) Collect pixel with the same binary response (CPP, Combinatorial Protein Pattern)

Binary Code-Table	CPP {A}	CPP {B}	CPP {C,D}
Antibody 1	0	0	0
Antibody 2	0	1	1
Antibody 3	1	0	1
Antibody 4	1	1	0
Frequency of CPP	1	1	2

(4) Introducing the concept of motifs (A Motif allows to look at protein subsets)

	Examples of Motifs	
Binary Code-Table	Motif derived from {{B},{C,D}}	Motif derived from {{A},{B}}
Antibody 1	0	*
Antibody 2	1	*
Antibody 3	*	*
Antibody 4	*	1
Frequency of Motif	3	2

* = Wildcard
1 = Antibody present
0 = Antibody not present

Figure 14: General procedure from MELK images to the definition of motifs.

7.2 MelTec's MotifFinder Technology

Since the detailed working process of MelTec's MotifFinder Technology has not been published yet, only a very brief overview can be given in this section. The MotifFinder is one key component of MelTec's data mining process, which consists of two different components: (1) the finding of 'interesting' motifs[14] and (2) a human data mining phase, where the results from (1) are reviewed and validated by a biologist. The goal of our approach is to test whether the results from the first step should pass through the second step – or not.

The MotifFinder analyzes the MELK-data (see Section 7.1). It searches through many motifs and produces an output of potentially interesting motifs. For n markers there are 3^n possible motifs. Several heuristics are used to reduce the search space. So, if the relative frequency of a certain motif is higher in one group than in the other, the motif is marked as 'interesting'. The difference is measured by statistical standard tests like Student's t-test and Wilcoxon Sum Rank test. The main advantages of such statistics are that the biological community is quite familiar with t-tests and p-values. These results are only intermediate results, which are explored, validated, and explained by a human expert.

7.3 Resampling Experiment

We applied the basic algorithm (see Section 5.3) to assess the multiple testing error probability also to MelTec's MotifFinder technology. The resampling procedure is implemented in Matlab and not in Perl5 (as for the experiments given in Chapter 6), since the MotifFinder (see Section 7.2) is a Matlab application. The principles for the Matlab implementation are the same as for the Perl5 implementation which is discussed in Section 6.1. So, the results can be compared directly. For the experiments with the MotifFinder the number of repetitions of the resampling procedure is set to 100, since the data analysis process is very time consuming due to the size of the data sets.

7.4 Experimental Result

In contrast to the data sets, which are used in Chapter 6, we used a not yet published data set – MelTec's TIMM data set. This data set provides a special case, since it is not known whether there is useful information contained in it or not. MelTec's TIMM data set is based on biological samples of two inflammatory diseases that are measured on MelTec's proprietary MELK technology. These diseases can be distinguished on eye inspection by a physician. One group consists of six samples from different patients and the other group consists of nine samples from different patients. Each sample is characterized by 19 protein markers. The aim of the MotifFinder analysis is to find so-called motifs (motifs are comparable to rules in association rule learning) that distinguish both diseases on a molecular level.

14 A motif is a representation of protein markers in a biological sample. Zeros are representing the absence and ones are representing the presence of a certain marker. Additionally, a motif allows wildcards (represented by asterisks). These wildcards are indicating that the corresponding marker does not contribute to a difference.

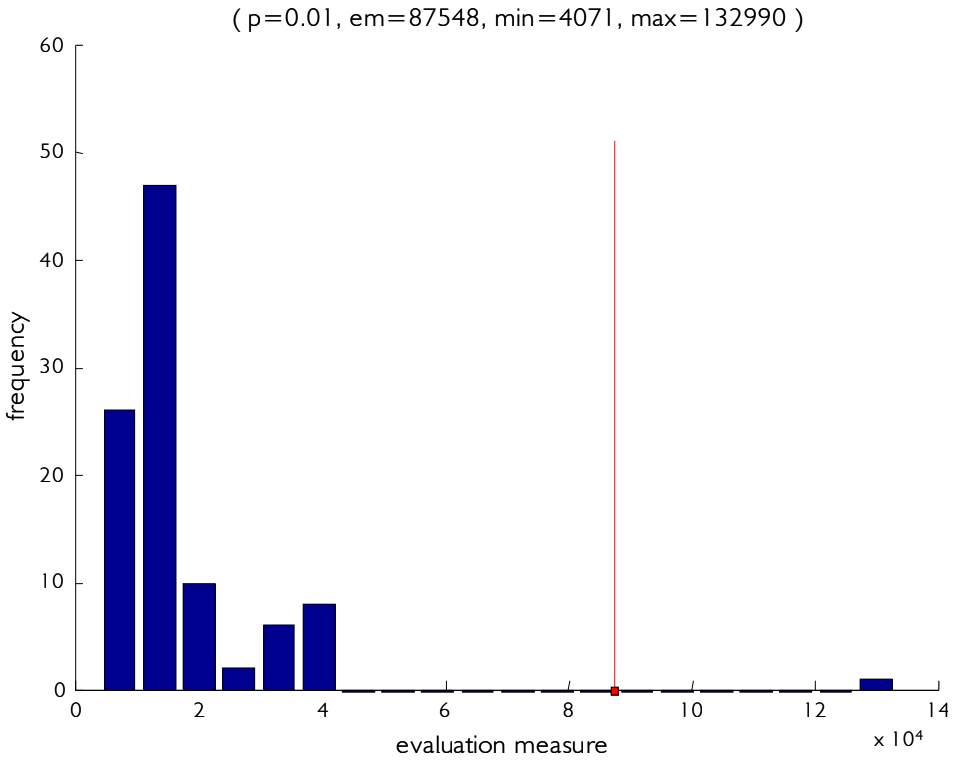


Figure 15: Resampling plot for MelTec's TIMM data set.

As one can see in Figure 15, there is quite a good separation between the resampling distribution and the original value. The resulting estimated p-value is quite small, so we conclude that the results for MelTec's TIMM data set must contain useful information. For the permuted data sets most values of the evaluation measure (the number of motifs) are conspicuous smaller than the value of the evaluation measure for the original TIMM data set – only for one permuted data set the value of the evaluation measure is larger. Hence, further experiments should be performed with new samples from new patients.

Chapter 8 – Conclusions

The approach to assess the multiple testing error, which is investigated by this thesis, is designed to deal with problems in which at least two different groups or classes exists, which should be distinguished by a data mining approach. By permuting the class labels it is possible for certain data mining approaches to compare the results of the original data set with results of data sets, which contain no (or less) useful information. This comparison is used to assess the multiple testing error. Our approach provides useful results and the p-value is a measurement which is commonly used in the bio-science community, so it is quite transparent for bio-scientists. The introduced approach does not provide such strong information about the probability of failure than other common approaches in the field of statistics (as for example the false discovery rate, FDR), since our approach does not evaluate each single hypothesis, which was tested by the data mining approach. But our approach can handle strongly dependent hypotheses. An advantageous feature of our approach is that it can be adopted for many data mining approaches without the need to change these standard data mining algorithms. An important fact that one should regard is that the selection of a suitable evaluation measure can become a crucial aspect – using a less suitable evaluation measure can lead to poor results; the selection of a suitable evaluation measure may change for different data sets. Actually, the resampling part is time-consuming, so it is necessary to use efficient data mining algorithms; but especially for the application at MeITec it is less expensive to run computer simulations than to run further experiments on biological samples. Fortunately, the result for MeITec's TIMM data set is promising, so this data set and the result, which is generated by the MotifFinder, should be investigated in more detail by the biologists.

Possible steps that could be seen as further work:

- test of suitable other numbers of resampling steps, especially fewer resampling steps
- examination of other data mining approaches and selection of other possible evaluation measures and evaluation of the behavior with pruned decision tree as data mining approach
- run especially the experiments with synthetically generated data sets with other data mining approaches
- more experiments with data sets with unknown information content
- instead of Perl5 scripts build a standalone application

Appendix A

The following code shows a Perl5 script, which was used to run the experiments with the census data set (see Section 6.2). It serves as an example for all other experiments which are done by similar scripts.

```
##############################################################################
# experiment setup for UCI_ML census data set ################################
##############################################################################

# administrative part for uci_ml_converter
my $CONVERTER_SCRIPT                = "converter.cmd";
my $UCI_ML_DATA_FILE                = "uci_ml_archive/census/census-income.combined";
my $CONVERTER_CONFIG                = "uci_ml_archive/census/config.txt";
my $CONVERTER_RESULT                = "user_data/census/census-income.converted";
my $CONVERTER_PERM_PATH             = "user_data/census/census-income/";

# administrative part for association rule learner
my $ASSOCIATION_LEARNER             = "apriori.exe";
my $ASSOCIATION_LEARNER_PARAMS      = "-ts -m3 -n7";
my $ASSOCIATION_LEARNER_INPUT       = $CONVERTER_RESULT;
my $ASSOCIATION_LEARNER_OUTPUT      = "user_data/census/census_ts_m3_n7.result";
my $ASSOCIATION_LEARNER_PERM_PATH   = "user_data/census/census_ts_m3_n7/";

my $FILTER_ATTRIBUTE                = "education=";
my $FILTER_RESULT_FILE              = "user_data/census/census_ts_m3_n7.filtered";

my $COUNT_FREQUENCIES_FILE          = "user_data/census/census_ts_m3_n7_freq.txt";


##############################################################################

print `mkdir $ASSOCIATION_LEARNER_PERM_PATH`;

# run converter
#START_CONVERTER();

# start learner
START_LEARNER();

# filter results
FILTER_RESULT();

# get number of results
CALC_RULE_NUMBER();

# same for resampled data
START_RESAMPLING_LEARNER();


##############################################################################
# function converts uci_ml dataset to input for mr. borgelt's tool
sub START_CONVERTER {
    print "\nSTARTING CONVERTER ... \n\n";
    print "$CONVERTER_SCRIPT $UCI_ML_DATA_FILE $CONVERTER_CONFIG $CONVERTER_RESULT \n\n";
    print `$CONVERTER_SCRIPT $UCI_ML_DATA_FILE $CONVERTER_CONFIG $CONVERTER_RESULT`;
    print "\n";
}

##############################################################################
# function starts mr. borgelt's tool with converted input
sub START_LEARNER {
    print "\nSTARTING ASSOCIATION LEARNER ... \n\n";
    print "$ASSOCIATION_LEARNER $ASSOCIATION_LEARNER_PARAMS $ASSOCIATION_LEARNER_INPUT
              $ASSOCIATION_LEARNER_OUTPUT \n\n";
    print `$ASSOCIATION_LEARNER $ASSOCIATION_LEARNER_PARAMS $ASSOCIATION_LEARNER_INPUT
              $ASSOCIATION_LEARNER_OUTPUT`;
    print "\n";
}
```

```
################################################################################
# function starts resampling procedure
sub START_RESAMPLING_LEARNER {
    my @PERM_FILES = FIND_FILES($CONVERTER_PERM_PATH);
    print @PERM_FILES;
    for ($i = 0; $i < @PERM_FILES; $i++) {
        # adjust file names
        $ASSOCIATION_LEARNER_INPUT = $CONVERTER_PERM_PATH . $PERM_FILES[$i];
        $ASSOCIATION_LEARNER_OUTPUT = $ASSOCIATION_LEARNER_PERM_PATH . "result_" .
             $PERM_FILES[$i];
        $FILTER_RESULT_FILE = $ASSOCIATION_LEARNER_PERM_PATH . "filtered_" . $PERM_FILES
             [$i];

        # start learner
        START_LEARNER();

        # filter results
        FILTER_RESULT();

        # get number of results
        CALC_RULE_NUMBER();

    }
}

################################################################################
# function filters results
sub FILTER_RESULT {
    print "\nFILTERING ASSOCIATION LEARNER RESULT ... \n\n";
    print "cat $ASSOCIATION_LEARNER_OUTPUT | grep $FILTER_ATTRIBUTE > $FILTER_RESULT_FILE
          \n\n";
    print `cat $ASSOCIATION_LEARNER_OUTPUT | grep $FILTER_ATTRIBUTE > $FILTER_RESULT_FILE`;
    print "\n";
}

################################################################################
# function count lines of the result file
sub CALC_RULE_NUMBER {
    print "\nNUMBER OF FILTERED ASSOCIATION LEARNER RESULTS ... \n\n";
    print `cat $FILTER_RESULT_FILE | wc --lines`;

    print `cat $FILTER_RESULT_FILE | wc --lines >> $COUNT_FREQUENCIES_FILE`;
}


################################################################################
# function lists files in a directory
sub FIND_FILES {

    my @FILE_NAMES;
    my $DIRECTORY = $_[0];

    opendir(DIR,$DIRECTORY);

    my $counter = 0;
    my $index = 0;

    while($FILE = readdir(DIR)) {
        if ($counter<2) {
            $counter = $counter + 1;
        }
        else {
            $FILE_NAMES[$index-2] = $FILE;
        }
        $index = $index + 1;
    }

    closedir(DIR);

    return @FILE_NAMES;
}
```

The following code serves as an example for a configuration file for the UCI-ML data set converter. It allows a conversion of the census data set (see Section 6.2).

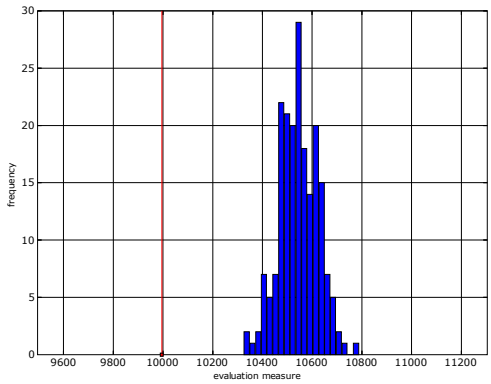
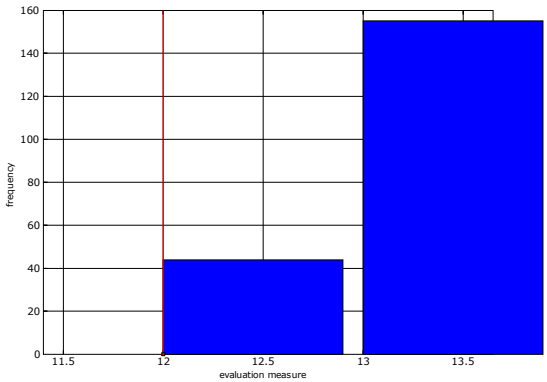
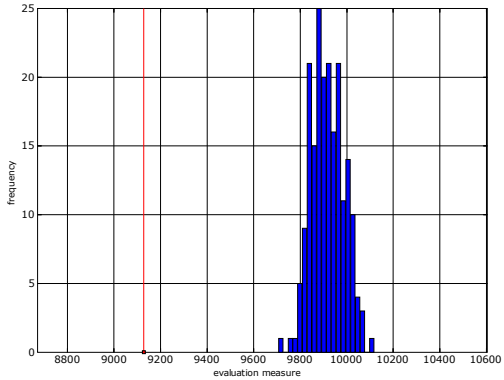
```
# first line:  attribute names
# second line: should this attribute be returned as output?
age,  workclass, fnlwgt, education, education-num, marital-status, occupation, relationship,
             race, sex,  capital-gain, capital-loss, hours-per-week, native-country, salary
true, true,      false,  true,      false,         true,           true,       true,
             true, true, true,         true,         true,           true,           true

# allows to split numerical attribute values into little groups
#   first:  attribute name
#   second: upper limit (<) for attribute value
#   third:  attribute value replacement
#   last:   default attribute value replacement
age, 26, young, 46, middle-aged, 66, senior, old
hours-per-week, 25, half-time, 41, full-time, 61, over-time, too-much
capital-gain, 2000, none, 8000, medium, high
capital-loss, 2000, none, 8000, medium, high

# stearing which (if any) attribute should be permuted and how often
perm, education, 200
```

The following code shows the replacement of certain attribute values, which is performed to generate the CensusHS-Induced data set (see Section 6.2). This replacement is performed with a certain probability. The first line defines the attribute, which value should be changed to the value, which is given in the following lines. If a replacement is performed all attribute values are changed to the values, which are specified by the corresponding line. To distinguish the 'induced' rules from other rules, we used attribute values, which are definitely not included in the original data set.

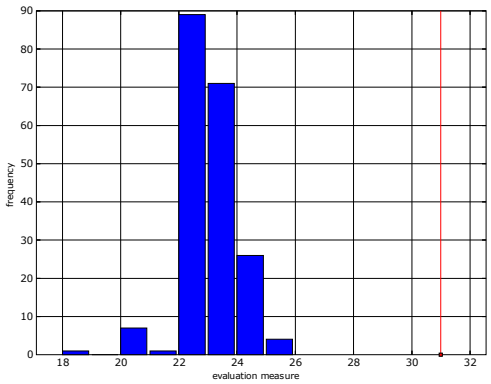
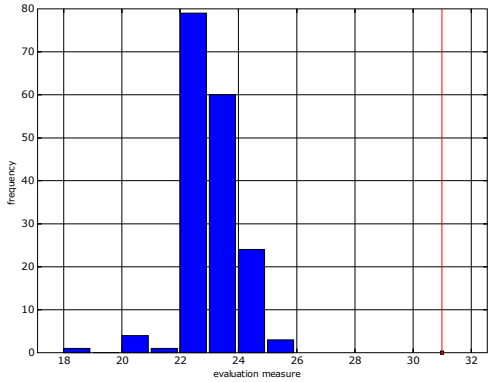
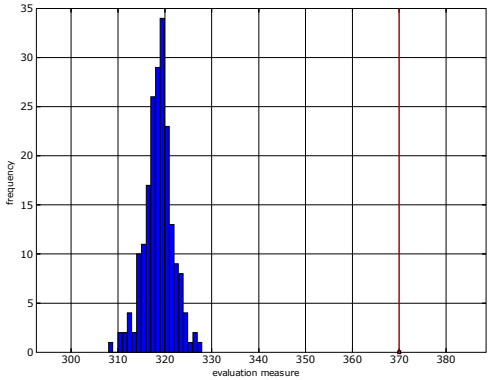
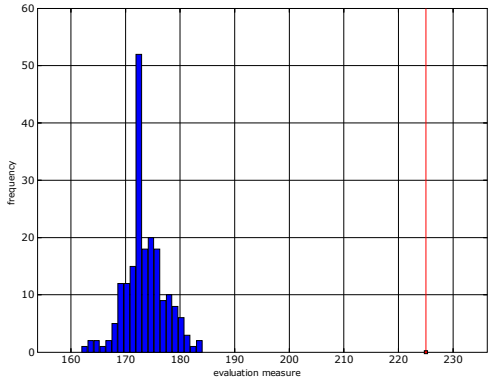
```
education, capital-gain, race,     native-country
HS-grad1,  60000,        vulcan,   Vulcan
HS-grad0,  100,          andorian, Andoria
HS-grad0,  6000,         romulan,  Romulus
HS-grad1,  100000,       human,    Earth
```

Appendix B

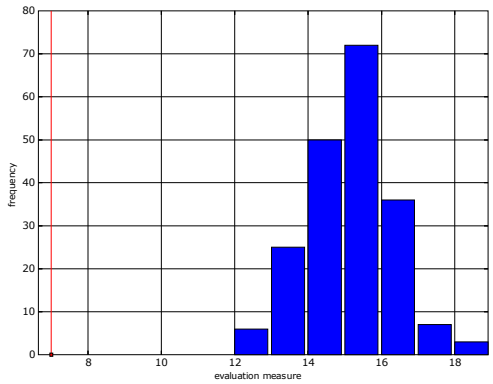
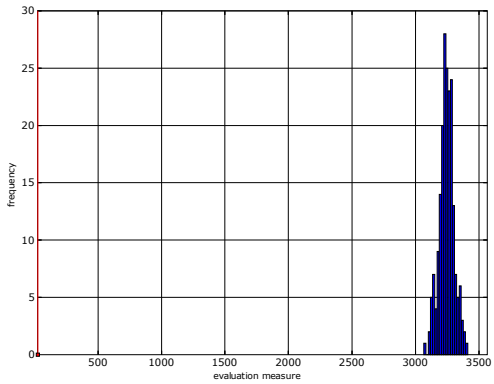
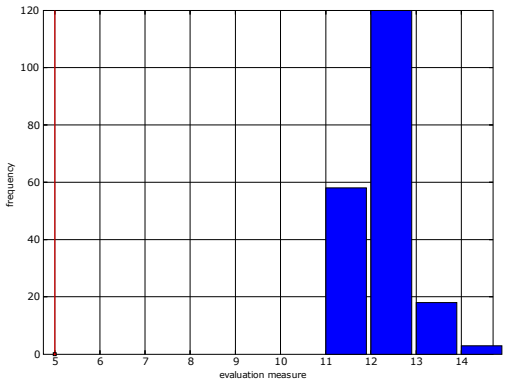
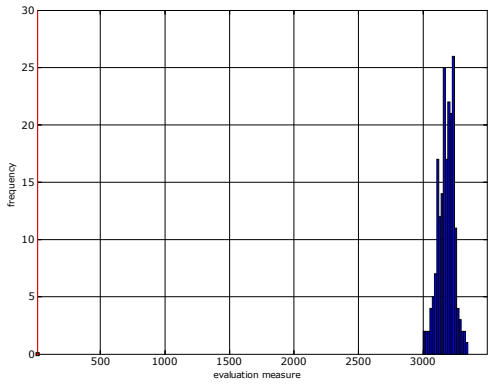
Census data set – Decision Trees

tree depth	**number of nodes**
with information gain ratio [no plot possible]	with information gain ratio  $p_{est.} < 0.005$
with information gain  $p_{est.} = 0.221$	with information gain  $p_{est.} < 0.005$

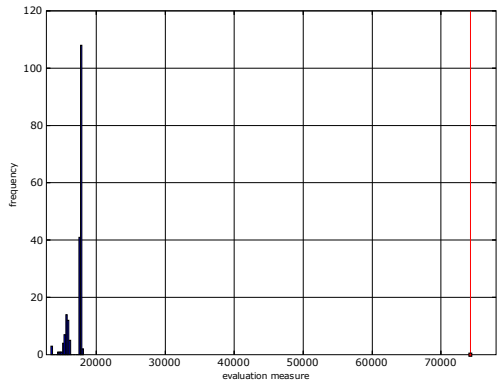
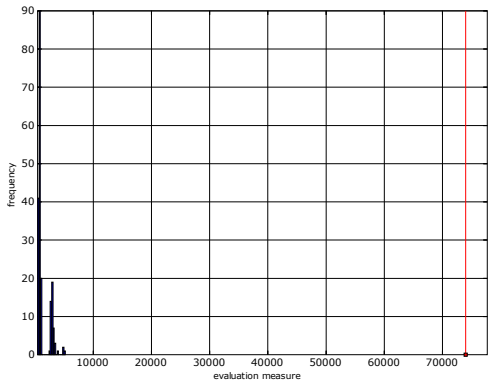
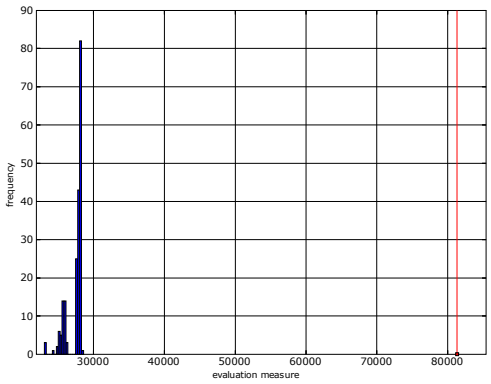
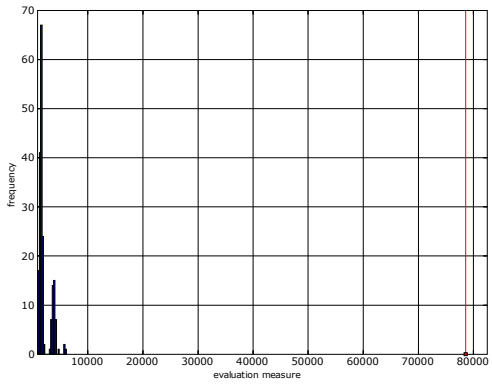
Census data set – Frequent Item Sets

number of rules with target attribute	number of non-ambiguous rules
with hyperedges	with hyperedges
	
$p_{est.} < 0.005$	$p_{est.} < 0.005$
with item sets	with item sets
	
$p_{est.} < 0.005$	$p_{est.} < 0.005$

Mushrooms data set – Decision Trees

tree depth	**number of nodes**
with information gain ratio	with information gain ratio
	
$p_{est.} < 0.005$	$p_{est.} < 0.005$
with information gain	with information gain
	
$p_{est.} < 0.005$	$p_{est.} < 0.005$

Mushrooms data set – Frequent Item Sets

number of rules with target attribute	number of non-ambiguous rules
with hyperedges  $p_{est.} < 0.005$	with hyperedges  $p_{est.} < 0.005$
with item sets  $p_{est.} < 0.005$	with item sets  $p_{est.} < 0.005$

Glossary

H_0	null hypothesis
H_A	alternative hypothesis
α	significance level
:=	is defined as
E[X]	expected value of random variable X
$\mathbb{R}$	space of real numbers
CER	comparison-wise error rate
EER	experiment-wise error rate
MEER	maximum experiment-wise error rate
FDR	false discovery rate
$\Rightarrow$	induced dependency
iff	if and only if
e.g.	for example
w.r.t.	with respect to

References

[Agrawal1993]: R. Agrawal, T. Imielienski, & A. Swami: Mining Association Rules between Sets of Items in Large Databases, Proc. Conf. on Management of Data, 207–216, ACM Press, New York, 1993.

[Agrawal1994]: R. Agrawal & R. Srikant: Fast Algorithms for Mining Association Rules, Proc. of the 20th VLDB Conf., Santiago, 1994.

[Bender2001]: R. Bender & S. Lange: Adjusting for multiple testing – when and how?, Journal of Clinical Epidemiology 54, Elsevier, 2001.

[Benjamini1995]: Y. Bejamini & Y. Hochberg: Controlling the False Discovery Rate – A Practical and Powerful Approach to Multiple Testing, JRSS-B 57: 289-300, 1995.

[Borgelt1998]: C. Borgelt & R. Kruse: Attributauswahlmaße für die Induktion von Entscheidungsbäumen – Ein Überblick, In: Gholamreza Nakhaeizadeh (Ed.) Data Mining – Theoretische Aspekte und Anwendungen, 77-98, Physica-Verlag, Heidelberg, 1998. (in German)

[Borgelt2002]: C. Borgelt & M.R. Berthold: Mining Molecular Fragments – Finding Relevant Substructures of Molecules, IEEE International Conference on Data Mining (ICDM 2002, Maebashi, Japan), 51-58, IEEE Press, Piscataway, 2002.

[Borgelt2005]: C. Borgelt: Free Implementations of Data Mining Algorithms, Otto-von-Guericke-Universität Magdeburg, 2005.
http://fuzzy.cs.uni-magdeburg.de/~borgelt/software.html

[Domingos1999]: P. Domingos: The Role of Occam's Razor in Knowledge Discovery, Data Mining and Knowledge Discovery 3(4), 409-425, 1999.

[Dudoit2003]: S. Dudoit, Y. Ge, & T.P. Speed: Resampling-based Multiple Testing for Microarray Data Analysis, Sociedad de Estadística e Investigación Operativa, Test Vol. 12, No. 1, 1–77, 2003.

[Efron1994]: B. Efron & R.J. Tibshirani: An Introduction to the Bootstrap, Chapman & Hall/CRC, 1994.

[FIMI2003]: B. Goethals & M.J. Zaki (Eds.): Frequent Itemset Mining Implementations 2003, Proc. of the Workshop on Frequent Itemset Mining Implementations (FIMI-03), Vol. 90, Melbourne, USA, 2003.

[FIMI2004]: R. Bayardo, B. Goethals, & M.J. Zaki (Eds.): Frequent Itemset Mining Implementations 2004, Proc. of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Vol. 126, Brighton, 2004.

[Fisher1935]: R.A. Fisher: The Design of Experiments, Third Edition, Oliver & Boyd Ltd., London, 1935.

[Hochberg1988]: Y. Hochberg: A sharper bonferroni procedure for multiple tests of significance, Biometrika 75, 800-802, 1988.

[Holm1979]: S. Holm: A simple sequentially rejective multiple test procedure, Scand. J. Statist. 6, 65-70, 1979.

[Manly2001]: B.F.J. Manly: Randomization, Bootstrap and Monte Carlo Methods in Biology, Second Edition, Chapman & Hall/CRC, 2001.

[Manly2004]: K.F. Manly, D. Nettleton, & J.T.G. Hwang: Genomics, Prior Probability, and Statistical Tests of Multiple Hypotheses, Genome Research 14, 997-1001, Cold Spring Harbor Lab. Press, 2004.

[Milton1990]: J.S. Milton & J.C. Arnold: Introduction to probability and Statistics – Principles and Applications for Engineering and the Computing Science, Second Edition, McGraw Hill Series in Probability and Statistics, 1990.

[Mitchell1997]: T.M. Mitchell: Machine Learning, McGraw-Hill, 1997.

[Pearl1991]: J. Pearl & T.S. Verma: A theory of inferred causation, Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning, Morgan Kaufmann, 441-452, San Mateo, 1991.

[Quinlan1993]: J.R. Quinlan: C4.5 – Programs for Machine Learning, Morgan Kaufmann, 1993.

[Storey2001]: J.D. Storey & R. Tibshirani: Estimating False Discovery Rates Under Dependence, with Applications to DNA Microarrays, Technical Report 2001-28, Dept. of Statistics, Stanford Univesity, 2001.

[UCI-ML1998]: S. Hettich, C.L. Blake, & C.J. Merz: UCI Repository of machine learning databases, University of California, Dept. of Information and Computer Science, Irvine, 1998.
http://www.ics.uci.edu/~mlearn/MLRepository.html

[Westfall1993]: P.H. Westfall & S.S. Young: Resampling-Based Multiple Testing – Examples and Methods for p-Value Adjustment, J. Wiley & Sons, New York, 1993.

[Wrobel2000]: S. Wrobel, K. Morik, & T. Joachims: Maschinelles Lernen und Data Mining. In: G. Görz, C. Rollinger & J. Schneeberger (Eds.), Handbuch der Künstlichen Intelligenz, 517-597, Oldenbourg-Verlag, 2000. (in German)

[Zaki1997]: M. Zaki, S. Parthasarathy, M. Ogihara, & W. Li: New Algorithms for Fast Discovery of Association Rules. Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining (KDD'97), 283–296. AAAI Press, Menlo Park, 1997.

Erklärung

Ich erkläre hiermit, dass ich diese Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen benutzt und die diesen Quellen wörtlich oder sinngemäß entnommenen Ausführungen als solche kenntlich gemacht habe.

Magdeburg, 28. Juli 2005