Silesian University of Technology
Faculty of Automatic Control, Electronics and Computer Science
Institute of Informatics

Doctor of Philosophy Dissertation

Sequential covering regression rule induction and optimization of regression rule-based data models

Adam Skowron

Supervisor: dr hab. Marek Sikora

Gliwice 2015

Contents

Acknowledgements

I would like to gratefully and sincerely thank to my supervisor for his guidance, understanding and patience. I would also like to thank Łukasz Wróbel for his participation in numerous experiments and long talks on various aspects of rule induction. Finally, and most importantly, I would like to thank my wife Danuta for all the support she has given me through all of these years.

This work was supported by the European Union from the European Social Fund (grant agreement number: UDA-POKL.04.01.01-00-106/09).

1. Introduction

For last twenty years the data mining methods have been widely used in many fields of human activity. This activity and the widespread availability of computers along with their growing disk storage led to the accumulation of vast amounts of diverse data that previously probably would have been omitted or discarded. The huge amount of data requires to develop newer and newer methods of data mining to gain valuable and useful knowledge. A relatively large computing power enables the development of increasingly sophisticated methods and thus helps in the knowledge extraction.

Data mining is commonly characterized as a multi-stage and mostly iterative process of extracting knowledge requiring the user to have not only the skill to use specific analytical methods, but also knowledge about a particular area of application. In the best case, the execution of the particular task of data mining is carried out by making a team consisting of an analyst and a domain expert.

To organize the data mining process and lead to its greater transparency, several methodologies have been developed. They describe the successive stages and relationships of the process. The most popular are: CRISP-DM, SEMMA, Six-Sigma and Virtuous Cycle of Data Mining [9]. In all of these methodologies one can find common phases. These are: defining the aim of the process, preparation and pre-processing of the data, modelling (that is the main stage of the process), model quality assessment, the interpretation of the results, and finally, the deployment phase that allows to use the model in a real-life process.

The most popular methods used in the modelling stage are: clustering, neural networks, support vector machines, tree induction and rule learning [132]. The consequence of the choice of the analytical method is different knowledge representation. The most understandable representation of human knowledge is generally considered to present it in the tree or rule representation. Due to the clarity of the knowledge representation, the tree and rule induction methods are commonly used to solve the problems where the readability is one of the most important factors of the model.

It is worth noting that description ability in rule induction is always important, even if rules are defined for the other purposes. For example, in classification, the clarity and readability of the model, rather than the classification accuracy, are a particularly accentuated feature of rule-based models [19, 80, 91]. Taking into account only the classification ability of the rule-based model, one could conclude that other methods (e.g. support vector machines, ensembles of classifiers) outperform rule-based models.

Good descriptive and classification performance has led various authors to apply rule induction in the survival analysis [130] or to solve regression problems [13, 30, 34, 51, 95, 120]. The latter is the main topic of this thesis.

For the first time the term *regression* was proposed in the 19^{th} century by Francis Galton [38] who dealt with genetics and eugenics. He observed that although tall parents have tall children the heights tend to regress down towards normal average. Nowadays the regression term is more general and describes the process of estimating the relationship between the dependent variable and the independent variables, also called explanatory.

Although previous studies on the problem of regression concerned a number of different approaches like linear regression, neural networks [113], support vector machines [49], transformation of the regression problem into the classification problem [120], learning regression rules from regression trees [13, 95] or based on ensemble techniques [30], as well as utilizing the most important for this work the so-called separate-and-conquer strategy [34], there are still open questions and areas to explore. There are some works on the topic of the separate-and-conquer strategy [57, 59] still some research areas have not been fully covered. For example, it is confirmed by numerous empirical research works that the heuristic used to control the induction process has a substantial impact on the final performance of the algorithm [4, 15, 16, 58, 102, 103, 110]. While some research on the heuristic in the regression rule induction has been undertaken [56, 58], there still remain promising heuristics well known in the classification, but not considered in the regression.

Improving the descriptive and predictive abilities can be also achieved through the use of techniques that can be generally called rule optimization techniques. In rule induction the rule optimization is performed in one of two stages: during or after the rule induction. However, in both cases, the rule optimization is most often associated with the so-called pruning. The main goal of these algorithms is to simplify or elimine unnecessary rules. The algorithms from the first group are then called pre-pruning algorithms while the algorithms from the second group- post-pruning algorithms. Moreover, the second group of optimization algorithms is independent of the induction algorithm. However, the research in the field of rule optimization algorithms concerns rather classification systems than their regression counterparts. In addition, the promising direction of research could be to investigate the use of different heuristics for the process of induction and the related pre-pruning algorithm.

1.1. Goals

The main goal of this work is to investigate and evaluate sequential covering rule induction and rule optimization algorithms for solving regression problems. The motivation behind this research is the existence of only few works on this topic [34, 57, 58, 59], which has many still untouched or insufficiently investigated fields.

Current studies in the field of regression rule induction mainly concern only the sequential covering rule induction algorithm running in the top-down strategy. The bottom-up strategy that works in the opposite direction, which in classification is presented as dedicated to imbalanced data [86], has not been examined yet in regression. It inclined us to develop the sequential covering rule induction algorithm with the use of the bottom-up strategy for regression. In addition we decided to investigate both algorithms and to introduce a modification of a fixed target value contributing to create quasi-covering algorithms. In addition, the above mentioned works checked only few from over 50 different quality measures [15, 36, 58, 110] that are used to control the process of decision rule induction. For some of them a tendency to lead to better results was observed [110]. The confirmation of these results in regression seems therefore an obvious consequence and is one of the sub-objectives of this work. To our knowledge, in regression there is also no statistical correction that could be applied to change the numbers of positive and negative examples change the quality of the rule and thus affecting subsequent steps of sequential rule induction algorithms and finally modifying the obtained model. From what we know, there is also a lack of research on rule optimization methods applied during and after the regression rule induction. Finally, with relation to the unordered set of rules which we used, different methods of conflict resolution have to be examined.

1.2. Contributions

The presented thesis comprises several contributions to the area of the regression rule induction. First, we have examined the top-down strategy and developed a new approach to the bottom-up strategy of sequential rule induction algorithms with appropriate modifications for regression. For both algorithms we have also applied two modifications of the fixed target value contributing to create quasi-covering algorithms.

Second, the induction process of the developed algorithms has been adapted, by an appropriate modification of the method of determining the positive and negative examples, to be under control of heuristics well-known from the classification rule induction. This application also enable us to traverse from the induction process control using only one heuristic to the separation of rule refinements and rule selection for regression with the use of separate heuristics.

The third contribution of this thesis is an application of rule optimization methods for regression. In the case of pre-pruning methods we have investigated the simplest Hill climbing pruning method and its modification which we called the Tabu hill climbing method (due to inspiration of a method already known as Tabu search [41, 42]). To optimize the rule-based model after induction we adapted for regression and examined six filtration algorithms: Inclusion, Coverage, Disjoint, Forward, Backward and ForwBack.

The fourth contribution is an examination of three (mean of conclusion, median of covered and max rule quality) and a proposition of one new method (mean of intersection) of resolving conflicts methods for the model in the form of an unordered set of regression rules.

The final contribution of this thesis are additional statistical corrections for the number of positive and negative examples which modifies regression rule quality using a given confidence level. This approach allows for dynamic modification of the rule towards optimistic or pessimistic rule evaluation possible due to assessment regarding population instead of the distribution of the sample.

1.3. Organization of the thesis

This thesis is structured as follows.

Chapter 2 introduces background information about rule-based data models. It gives a brief overview about an evaluation of a single rule with presentation of heuristics, about approaches for rule set evaluation as well as about statistical comparison between different regression models. Moreover, Section 2.5 presents four resolving conflicts methods including one originally introduced in this work.

Chapter 3 describes the first part of the main goal of this thesis presenting two different strategies (Top-down and Bottom-up) for regression rule induction and their modifications with the fixed target value in a rule conclusion. The chapter ends with the introduction to the rule quality evaluation using confidence intervals.

Chapter 4 presents the second part of the main goal of this thesis. It starts with a brief overview and general motivation behind rule optimization. Then the pre-pruning algorithms are presented, each in a separate subsection. The chapter is completed with the presentation of algorithms for filtration of regression rules.

In Chapter 5 the results of the extensive empirical evaluation on many diverse data sets are shown. In each section the focus lies in the evaluation of one aforementioned problem. At the end of the chapter the comparison of the best combination of methods to the state-of-the-art algorithms is given.

Chapter 6 presents the results of experiments on real-life data. The main goal of this chapter is to demonstrate that the presented algorithms, heuristics, methods and approaches can be used to solve authentic regression problems and, perhaps, to contribute to their commercial use.

Chapter 7 concludes with a summary of this thesis and provides direction for future work.

2. Rule-based data models

A rule induction is a branch of machine learning. In literature one can find many definitions of machine learning. One of the most popular and the most frequently quoted definition has been proposed by Tom Mitchell [84]. In simplified terms, this definition is as follows: *a computer program is learnt, if a performance of the program in solving a given problem, measured by a some performance measure, increases with the experience*.

A less formal, but more understandable, description of machine learning has been proposed by Ryszard Michalski [81]. According to him, the idea of machine learning concerns the process of incorporation of well-known capabilities of learning such as: acquisition of declarative data, development of skills through guidelines or practice, organization of knowledge in a general way, human-readable representation and discovery of facts or patterns based on observation and experiment, in computers. Moreover, both definitions boil down to the description of a system, which is commonly called a learning system. The main feature of such a system is the possibility to change its internal parameters in order to identify and describe the data.

Generally, machine learning is assigned to the area of artificial intelligence. However, this classification is not accurate. This is because the branches of science are not defined, but slowly form during the process of clustering the common objects and purposes of the study [26]. Much more accurate classification of machine learning is to assign it to the branch of computer science, which is computational intelligence [26]. Using this classification, one can say that machine learning examines the problems for which there are no effective computational algorithms.

Among the algorithms that fit under this definition, one can distinguish the groups of algorithms that are divided according to the way of learning. It is possible to mention the following ways of learning: based on examples, by memorizing, by analogy, based on the queries and with the gain. Learning based on examples is known as *induction*. However, the most common is traditional division of algorithms based on the availability of training information. Here one could specify supervised and unsupervised learning [10, 19, 33, 81, 127].

Supervised learning, using examples, consists of finding one or more hypotheses that describe certain concepts, classes or categories. The terms *concept, class* and *category* shall be understood as a set of examples that has some common and characteristic properties, which distinguish this group of examples from groups described by other concepts. Roughly it can be also understood as the result of the learning process, regardless of the type of learning [127]. The division of examples may be compared to the well-know representation of the binary logic. The elements that are instances of the given concept are called positive examples. The other examples are referred to as negative [10, 33].

Formulating hypotheses for machine learning systems should be also considered in relation to the demands presented by Michalski [80, 81]. The author suggested that the representation of hypotheses must satisfy the principle of intelligibility. This means that the description of the concepts using the hypotheses should be written in a manner understandable to humans. It is important to facilitate human understanding not only of the final results, but also the assumptions, principles and theories behind them [80].

These considerations lead to yet another classification of algorithms based on the method of knowledge representation. This group contains methods and algorithms which store information in a symbolic form, often using some strings, words or inscriptions [10, 19]. The knowledge saved in this way is human readable and hence more understandable. The second group includes methods which present the knowledge with the use of a numerical or more complex form, for example, binary strings. Such knowledge is not immediately understandable to humans and requires additional information or familiarity of assumptions. Generally, such algorithms are called non-symbolic or subsymbolic learning methods [10, 19].

Using the above classification one cay say that the group of symbolic methods, could comprise, inter alia, the representation using rules, graphs including decision trees or first-order logic. On the contrary, in the group of non-symbolic methods the following can be distinguished: neural networks, fuzzy sets, statistic methods based on probabilities or the traditional approach of evolutionary and genetic algorithms [19, 55].

Cichosz noted that the representation of knowledge can suggest, but does not clearly state, how this information will be used. However, the usage is determined on the basis of both the representation and the purpose for which this knowledge is obtained [19]. Among the most popular tasks one can mention classification and approximation (including regression). Equal importance should be assigned to the purpose for which the system simply presents to its user the readable knowledge which allows him to make use of it [19].

2.1. Foundations of data representation

Many machine learning algorithms are widely applied in the area of data mining, which mostly refers to the extraction of knowledge and / or interesting patterns. However, understanding the data representation is often more important than the learning process itself [127]. To facilitate the understanding of the input a large number of problems of data mining and knowledge discovery are represented by data in a tabular form. This simple and transparent data structure is also known as attribute-value representation [33, 127, 137] or a matrix of instances versus attributes [127]. The formal definition of such a tabular form is presented in Definition 1.

Definition 1. Let U be a set and $e \in U$ be an instance. The homogeneous finite set of instances e is called $universe$ and it is denoted as

$$U = \{e_1, e_2, ..., e_n\}\,. \tag{2.1}$$

The instance $e \in U$ that alternately takes the name of an object or an example is represented by the finite number of features, referred to as attributes. Each of attributes $a \in A$ is a function $a_i : U \rightarrow Va$, assigning to each object from the set U a certain value belonging to the set Va. The Va is an attribute range for the specified attribute a. Therefore, each instance can takes the form of

$$A(e) = \{a_1(e), a_2(e), ..., a_k(e)\}\,. \tag{2.2}$$

Attribute values for a specified instance represent a quantity measurement of a particular attribute and present information from observation, sensors, etc. Generally, the most common are two types of attributes: $numeric$ and $nominal$ ones (although one can also find types such as: ordinal, ratio, interval etc.). $Numeric$ attributes, which mostly reflect measurements, take real numbersas values. In turn, $nominal$ attributes can have a finite set of values, therefore sometimes they are called $categorical$. The nominal attributes have some special features worth mentioning. The values of such attributes serve as labels or names. Consequently, they cannot be ordered or measured by distance. Moreover, their values cannot be multiplied or added. They can be exclusively compared using the test for equality or inequality [127]. Whereas numeric attributes could be compared using mathematical relations such as $=$, $<$, $>$, $\leqslant$ or $\geqslant$, which will be discussed in details in the next section.

In machine learning systems there are also other types of attributes such as ordinal, interval, ratio, metadata (data about data), etc [127]. However, these attributes are less popular, and what is perhaps more important, it is sometimes hard to compare their values or differences between these values [127]. A good example of the difficulty in comparing attribute values may be the ordinal attribute $size$ with values: big, $medium$ and $small$. This attribute can be ordered $big > medium > small$, however it is not possible to measure the difference between big and $medium$ with respect to the difference between $medium$ and $small$. Witten, Frank and Hall emphasize that *"distinction between nominal and ordinal quantities is not always straightforward and obvious"* [127].

Apart from the types of attributes, all attributes can be divided into two disjoint subsets: *condition attributes* C and *decision attributes* D [90]. The attributes from the first group are also known as *independent variables* while decision attributes are called *dependent variables*. By definition, both subsets may contain a certain finite number of attributes. However, in real systems a decision subset has typically only one attribute. In the case of a collection of attributes in the decision subset such an array can be reduced to a single-element array where each decision is represented by a unique pair of replaced elements. The set of attributes A could be denoted

$$A = C \cup D. \tag{2.3}$$

Introduced symbols U (2.1) and A (2.2) can be used now to present the knowledge representation system in the form of a decision table [127]. This table input form was originally proposed in the rough set theory [90], however a general idea may also be used in other machine learning problems.

Definition 2. The decision table DT is a pair (U, A) where U is a finite set of examples and A is a finite set of attributes.

In addition, the decision table can be presented with the use of aformentioned condition and decision disjoints. The form of such a decision table will be denoted then

$$S = (U, A, C, D) \tag{2.4}$$

where S is the system, U is the universe, A is the set of attributes and C and D are the sets of conditional and decision attributes respectively. Foregoing considerations can be easily transferred to a regression, in which, instead of the symbolic decision attributes or groups of values, there are numeric decision attributes. Other considerations are the same as above.

As previously stated, the learning process should lead to some kind of knowledge. Furthermore, this knowledge should be unambiguous. This means that a particular decision should clearly result from the knowledge (patterns) hidden in the values of conditional attributes. If data lead to the opposite decision, then the result cannot be unambiguous. This problem can be illustrated by a simple example. Before turning to the example, it is worth mentioning that generally in the attribute-value representation the instances are the rows and the attributes are the columns and this notation will be preserved in this work. Then, assume that the decision table looks like Table 2.1 where attributes a, b and c are condition attributes

Table 2.1: Example of inconsistent regression table

U	a	b	c	d
1	1	0	2	3
2	2	1	1	1
3	1	3	1	1.5
4	1	0	2	5
5	1	3	1	1.5

and d is a decision attribute. As it can been seen, two examples 1 and 4 have identical structures $e_1 = e_4 = (1, 0, 2)$ but different decisions d: $a_d(1) = 3$ and $a_d(4) = 5$.

Formally, the decision table is inconsistent if two or more instances have all condition attributes identical but different decisions. Otherwise, the decision table is consistent [90].

To deal with the problem of the inconsistent table Pawlak has proposed to decompose the inconsistent table into two tables: the first one that is consistent and the second one that could be inconsistent [90]. In other words, the simplest and issuing a reasonable method is to remove inconsistent examples from the learning process. Therefore, the learning process should concern exclusively the consistent examples or involves more sophisticated methods like rough sets [112].

So far, attention has been paid to the general and formal division of attributes for condition and decision attributes. However, the issue of the decision attribute is more complex and crucial from the point of view of this work. The key is that the type of the decision attribute determines the type of the problem that should be solved. In the classification the algorithms predict the nominal or ordinal value of the target attribute [33, 45, 127]. In other words, the decision informs about the assignments of the examples to the specified class and in the particular case one has to take some action when the decision is one of the two classes: take action or not.

In this work the attention is focused on the problem of regression, where the target value is a numeric type, like in the example from Table 2.1 and the main goal is to predict this continuous value (also called as numerical target or regression value) [33, 120, 127]. Nonetheless, the presented properties for the system, universe, examples and attributes are the same for the classification and the regression. It is also worth noting that in case of regression the decision table is often a reference to the regression table [127].

2.2. Rule representation

The rule representation is one of the most popular [34, 33, 47, 69, 102, 116, 127, 128] and the most transparent and understandable forms to humans [19, 30, 51, 94, 110, 127].

Each rule takes the form of:

$$\textit{IF } \varphi \textit{ THEN } \psi \tag{2.5}$$

where φ is a $body$ and ψ is a $head$ of the rule. Therefore, a generic form of the rule is sometimes written as follows:

$$\textit{body} \rightarrow \textit{head} \tag{2.6}$$

and is read: "if $body$ then $head$". The Rule 2.5 could be also written in the generic form as $\varphi \rightarrow \psi$.

The condition part of the rule φ is a logical expression of some features. Whereas the conclusion ψ determines the type of the rule. There are many types of rules depending on the conclusion type. If the conclusion takes a form of logic expression than such a rule is called $logic\ rule$. If the rule contains in the conclusion some kind of a $decision$, then such rules are called $decision\ rules$ [10, 24, 103]. A few authors also proposed to use the term $classification$ $rules$ when decision rules are used to solve the classification problem [10, 24, 33, 75, 94]. There are also other types of rules, such as: $association\ rules$ [2, 10, 33, 87, 75] or $inhibitory\ rules$ [23].

The main objective of the association rules, which are very popular mainly in an area of e-commerce, is to study the impact of purchase of one product to another [2, 100, 122]. In turn, the inhibitory rules, in contrast to standard form of a rule, (Formula 2.5) have a form of *IF* φ *THEN NOT* ψ. In other words, the inhibitory rule implies the exclusion of some conclusions in the presence of a specific logical expression in the body.

The most interesting from this point of view are the so-called $regression\ rules$ [13, 24, 58, 59, 95, 120, 125]. In the regression rules the conclusion is a numerical type, so the examples that meet all the conditions receive a predicted continuous value.

The regression rules are defined for descriptive and predictive purposes. For the descriptive perspective the most interesting would be the set composed of rules presenting the non-trivial and useful information to the user. For the prediction perspective, the most desirable would be the set composed of rules that allows to obtain the most accurate prediction of the *value of dependent variable* based on the information from the *values of independent variables*. It should be also emphasized that in all rule induction tasks the descriptive ability is important.

It is the clarity of the data model, rather than the efficiency of classification or prediction, that is a particularly accentuated feature of rule-based models [19, 80, 91]. Considering only the accuracy of rule-based models, one comes to the conclusion that many other methods (e.g. support vector machines, neuro-fuzzy systems, ensembles of classifiers) outperform rule-based models.

As it was mentioned above, the condition part of a rule consists of a logical expression. A more specific definition says that in the case of decision or regression rules the *body* of a single rule is a logical conjunction of conditions, where each condition checks the fulfillment of a given property. A transformation of the general form of rules into conjunction of features can be written as follows:

$$\textit{IF}\ \ w_1 \wedge w_2 \wedge \ldots \wedge w_j\ \ \textit{THEN}\ \ \psi \tag{2.7}$$

where each w is an elementary condition of a given rule. In real systems the number of elementary conditions is finite and is defined as $rule\ length$ [33].

In the regression problem, the aforementioned rule could be also written with the use of a substitution of ψ with a general notation of a function:

$$\textit{IF}\ \ w_1 \wedge w_2 \wedge \ldots \wedge w_j\ \ \textit{THEN f(x)} \tag{2.8}$$

where $f(x)$ in the simplest form is a single value obtained from all examples covered by the fired rule. Such an approach has, of course, its advantages and disadvantages. The simplest form of conclusion is primarily the most understandable and transparent way of present the prediction. This is also the fastest possible way of the prediction. Alternatively, the implementation of a linear model in the form of $w_0 + w_1a_1 + \ldots + w_ka_k$ is based on the values of many of the attributes (k) multiplied by some weights (w) and usually allows to obtain a smaller prediction error [51, 56]. However, it is evident that the clarity of rules with linear models decreases drastically. Due to the simpler form of a single value which also leads to more understandable interpretation, most of the results in this study will be presented in this form. However, to confirm the above-mentioned advantages of the linear model in a few experiments, additional results will be presented, but will be returned to later.

The fulfillment of each condition involves completing a logical expression inside it. In the standard definition, a rule covers an example if the condition part of the rule is met for this example. On the contrary, the examples that fulfill all conditions from one rule support this rule. Thus, there are two terms that describe the relation between the rule and the examples. The rule can cover examples and the examples may support the rule. The target value for the covered examples is assigned by the supported rule or, in some cases that will be discussed later, rules.

The single elementary condition can be generally written as $a \; op \; Va$. In this study, it was assumed that a stands for a specific attribute name, op is one of relation symbol from the set $\{=, \neq, <, >, \leqslant, \geqslant\}$ and Va is a numerical or nominal value from the range of the attribute a. The type of value Va depends on the type of the particular attribute a.

A sample rule for the dataset presented in Table 2.1 can be built as below:

$$\textit{IF} \;\; a \leqslant 1 \wedge c = 1 \;\; \textit{THEN} \;\; 1.5$$

However, it should be noted that the different algorithms can induce a different rule that covers the same subset of examples and implies the same conclusion. The produced rule can be bigger or smaller - in terms of $rule\ length$ - or it can consist of completely different attributes, e.g.

$$\textit{IF} \;\; b = 3 \;\; \textit{THEN} \;\; 1.5.$$

In those examples both rules point to the same subset of elements (e_3 and e_5) but in different ways.

In most cases one rule is not enough to cover all examples from the dataset. It rather covers only a smaller subset, in fact, a few or several examples. It can be seen that the rule presented above covers only 2 of 5 examples from Table 2.1. To cover the whole dataset, where each rule covers a part of whole dataset, more rules are needed. In literature, such a collection of rules is called a *rule set* [20, 33, 48, 127].

The rule set may be unordered or ordered (decision list). The main difference between ordered and unordered rules lies in assigning the target value. In the case of ordered rules the rules are checked one by one in a specified established order. Then the target value of the first rule, that covers the tested example, is assigned. For unordered rules, more than one rule may cover the tested example. Hence determining the target value is more difficult. The appearance of more than one covering rule may also occur in the other case.

Depending on the method of rule induction, the rule set may consist of *overlapping* or *non-overlapping* rules. The non-overlapping rules are the result of the application of an algorithm that can divide the dataset into completely separated subsets, e.g. regression trees, which can be further transformed to the form of rules. Alternatively, the rule learning algorithms induce rules that can overlap each other. This more relaxing approach often contributes to induce smaller rule sets [33].

2.3. Rule induction algorithms

The rule induction is one of the most popular methods of learning by examples. The rule induction algorithms, in a variety of approaches in classification or regression problems, try to tackle the problem of the rule set production, which can be compared or characterized by for example: the best classification accuracy, the smallest prediction error in regression, the best descriptive ability etc. However, the two main strategies of automatic rule induction are Divide-and-conquer and Separate-and-conquer [12, 34, 68, 93, 94, 118, 127].

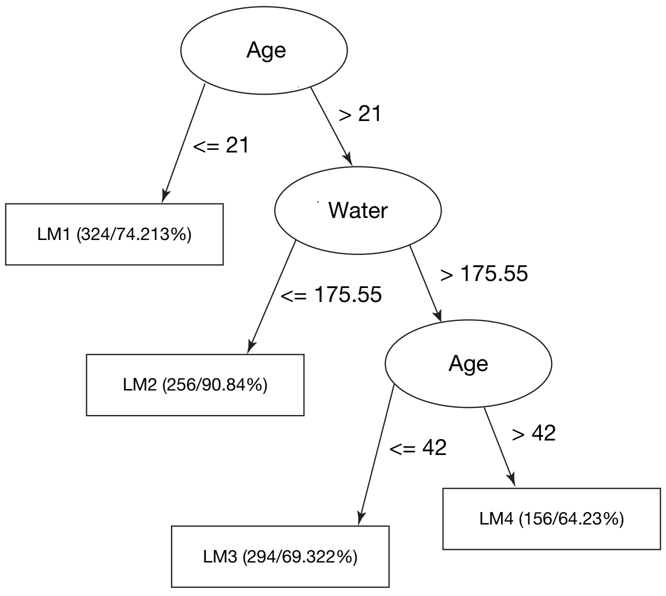
In the production process of a rule set the Divide-and-conquer algorithms formulate hypotheses by splitting the most general rule to the specialized rules. The process starts from the most general rule. Subsequently, based on the assumptions of an algorithm, one best attribute is selected, which can divide example set into two subsets. If at least one subset contains only positive examples, then the process is stopped for this subset. Otherwise, the process is recursively repeated until all examples belong to disjoint subsets with only positive instances inside [12, 94, 118].

Equally popular is an explanation of the Divide-and-conquer process with respect to the *branches* and *nodes*. In this definition the initial attribute is denoted as a root node. Then the algorithm makes a branch for the selected attribute to split the dataset into two subsets. The splitting process is repeated until examples do not belong to the same class (for classification) or are outside of the target value range (for regression). Otherwise, the process is repeated until the division can be performed [118, 127]. Therefore, considering the analogy of the tree construction, the algorithms from this group are called tree-based.

The Divide-and-conquer strategy has been mostly developed and improved over many years by J. Ross Quinlan [93, 94, 95, 96, 97, 127]. Although his approach, called $ID3$, has been proposed to build decision trees, instead of rules, it can be regarded as one of the most important works for the development of this scheme [93, 127]. A collection of improvements appeared later in a practical and reliable system $C4.5$ that had and still has a huge impact on the creation of new algorithms in many of machine learning areas [7, 34, 86, 96, 127]. The commercial successor of $C4.5$ is $C5.0$ with few improvements e.g. in speed, memory usage or size of the produced decision tree [76].

From the regression point of view, the works of Breiman et al. ($CART$ algorithm) [13] and Quinlan ($M5$ algorithm) [95] are particularly important. In both cases the authors have proposed to convert the decision at leaves into the predicted numerical target value ($CART$) or a multivariate linear models ($M5$). Apart from the implementation details of both algorithms, in which, however, $M5$ produced smaller model trees, in both cases the general idea of creating models is similar [95]. It is also worth noting that the first implementation of the M5 algorithm has been described in a very general way and the strategy has been improved by Wang and Witten in a system called $M5'$ [51, 125]. In this system, among many changes, one of the most important is that the heuristics used to split examples into disjoint subsets has been replaced by a measure to minimise the intra-subset variation [13, 51, 59, 95, 134].

By proceeding sequentially from decision trees through regression trees, a set of regression rules can be finally obtained. Such an approach has been presented by Holmes et al. in a system called $M5Rules$. The tree construction is done with the use of the $M5'$ algorithm. Then the rule is generated using the best (according to a certain heuristic rule) leaf. The rule body is built based on all attributes along the path from the best leaf to the root. In the last step of creating a single rule all examples covered by that rule are removed from the dataset and the process is recursively repeated until no instances remain [51].



Linear models:

LM1: 26.4411 - 0.0143 * Water + 0.0052 * Age

LM2: 54.5883 - 0.0243 * Water + 0.0062 * Age

LM3: 35.5201 - 0.017 * Water + 0.0067 * Age

LM4: 44.3338 - 0.017 * Water + 0.0083 * Age

Figure 2.1: An output visualization of M5’ algorithm

Figure 2.1 presents a tree produced by the $M5'$ algorithm for a reduced set of attributes (Age and $Water$ only) from the real dataset $compressive$. Each branch shows the values of the attribute in which the set is split into two disjoint subsets while the leaves display the linear model, the coverage and the percent root mean squared error respectively. A set of rules generated by the algorithm $M5Rules$ for exactly the same values of the parameters is as follows:

IF $Age \leqslant 21$ *THEN* $26.4411 - 0.0143 \cdot Water + 0.0052 \cdot Age$

IF $Water > 175.55$ *and* $Age \leqslant 42$ *THEN* $34.3821 - 0.0105 \cdot Water + 0.0043 \cdot Age$

IF $Water \leqslant 189.145$ *THEN* $52.7158 - 0.0088 \cdot Water$

OTHERWISE 40.5694

It is noteworthy that only the first rule has its counterpart in the model tree in the linear model 1. It can be also noted that in the rule set there is a default rule for instances not covered by any of the generated rules. In *M5Rules* the value of the default rule is calculated as an average of the target values of all examples belonging to the training set. It is worth mentioning that there are more sophisticated methods of determining the default target value such as linear models or k-means clustering. However, they are also more difficult to interpret.

Conversely, the Separate-and-conquer approach works in a slightly different way. One of the best and the most frequently quoted presentations of this strategy has been proposed by Johannes Fürnkranz [34] who noted that this strategy had its origins in Michalski's work [77]. The general idea is that all algorithms belonging to the separate-and-conquer group operate in a looped manner. The general outline of the algorithm can be represented by Algorithm 1. The algorithm is looking for a rule that covers a part of training examples (the conquer part) and then covered instances are removed from the dataset (the separate part). This step is repeated as long as the training set has uncovered examples [34]. The algorithms that belong to this group, due to the rules that cover subset of examples, are also called *covering* algorithms.

Algorithm 1 Pseudocode of the *covering* induction algorithm

```
Input: examples - training set of examples
Output: ruleSet - set of induced rules
$ruleSet \leftarrow \emptyset$
while $examples \neq \emptyset$ do
    $rule \leftarrow FindRule(examples)$
    if $rule\ exists$ then
        $covered \leftarrow Covered(rule, examples)$
        $examples \leftarrow examples \setminus covered$
        $ruleSet \leftarrow ruleSet \cup \{rule\}$
    else
        break
    end if
end while
return $ruleSet$
```

Although the main loop (also the so-called top-level loop) of Separate-and-conquer algorithms is uniform for them, the method of induction may vary significantly for each single rule. Thus Fürnkranz pointed that each approach can be characterized with biases, which are used for these purposes. Depending on the source, one may mention three or four points [34, 35, 89, 127] that are used to differ algorithms. Here, the points are limited to three as in [34, 35], however the location of the dispute is underlined.

Language Bias

Language bias can be understood as opportunities and constraints arising and strongly dependent on the adopted form of representation hypotheses. Then the chosen form of representation affects the search space for a learning algorithm. However, the adopted language may not be sufficient to demonstrate all the concepts. In a simplified term the existence of one universal language would describe all possible divisions of examples and all concepts could be learned [127]. Witten et al. have pointed out that it is a rather theoretical consideration because in practice the problem is typically too large to show all the concepts using only one form [127]. The proposed solution is to separate concepts and describe them in a simpler form. The straightforward representation of hypotheses is also the fulfilment of the principles of intelligibility introduced by Michalski [80, 81].

Search Bias

The way of searching through the search space is one of the most characteristic features that differentiate algorithms from each other. After determining the manner of representation of hypotheses it is necessary to determine the search algorithm (usually hill-climbing, beam search or best-first, which could guarantee that an optimal solution will be found [34]), its strategy (top-down, bottom-up or bi-directional), which is also described asa higher-level of search bias [127], and the search heuristics. Pappa and Freitas have proposed to isolate the heuristics to a separate point as an evaluation measure of the searching result [89].

Overfitting Avoidance Bias

There are many algorithms that use some kind of safety mechanisms to handle noisy data or to avoid that the model has become too powerful (overfitting). This mechanism can lead to the more general model in hope that simpler hypotheses will provide higher accuracy on unseen examples [32, 34]. It is also worth mentioning that the easiest way to obtain an accurate and reliable theory is by simplifying the complex one [127]. Generally there are two families of methods dealing with the overfitting problem. The first approach is popularly called *pre-pruning* because the complex concept is pruned during the induction process. In turn, the second group is known as *post-pruning* because the theory is examined after the completion of its creation.

Although both methods have undoubtedly their pros and cons, the natural consequence shall be a combination of methods that would complement each other [32].

2.3.1. Related work

The origin of the *covering* strategy has its place in the classification algorithms. The first of these algorithms was proposed by Michalski in 1969, the AQ algorithm [77]. In the next years there were a number of modification of the basic algorithm leading to the creation of the whole family of AQ algorithms (e.g. AQ15, AQ17, AQ18, AQ19, AQ21) [11, 63, 79, 82, 83, 129]. The induction method for a single rule in all the programs is similar. The AQ family works in the top-down hill-climbing approach and uses the beam search method. The main idea of the basic AQ algorithm is to increase the coverage of a training set in each iteration. This coverage is provided by a set of accurate rules in which the elementary conditions are linked with logical conjunction, however an internal disjunction within one elementary condition is also allowed. In the modifications of the original AQ algorithm the inaccurate rules are also admitted.

Other very popular algorithm used for solving the classification problem is CN2 [20, 21], which is a modification of the AQ algorithm. The main difference between the CN2 and AQ algorithms is to extend the search space in such a way that rules could also cover negative examples. The other modifications are the use of an ordered list and a statistical evaluation of new elementary conditions (called here complexes) to check whether the complex is statistically significant or not. In the next version of the algorithm modificationshave been proposed to change the rule evaluation heuristics in order to prevent the occurrence of very specific rules that cover only few examples [20].

RIPPER is another algorithm worth mentioning due to its popularity. It uses the covering approach with the hill-climbing strategy. In the rule growth process RIPPER applies an information gain criterion to repeatedly add conditions until the rule covers no negative examples. The most differentiating feature of this algorithm from those two mentioned above is that the rule construction starts from the least prevalent class [22]. The *Separate-and-conquer* approach is also employed in such classification algorithms like IREP [37] or PART [28].

In the regression problem there are only few attempts to use *the covering* technique. All of them are briefly outlined below. It is interesting that each of the below works come to the topic of regression in a slightly different way. These approaches will be accentuated in each of the algorithms.

Karalič and Bratko have defined the FOR (First Order Regression) approach to handle numerical information in Inductive Logic Programming (ILP), which can be defined as a subfield of machine learning where the background knowledge is taken into account to create a hypothesis. This idea has been implemented in the program called FORS [62]. The algorithm starts from the most general rule (empty rule) and then it is specialized by adding clauses. To find the clause, the beam search is applied that searches the space of all possible clauses. Each clause candidate is evaluated based on *the mean squared error* estimator, which will be discussed later.

The specialization of a single clause is limited to three steps. First, the specialization can be done using the background knowledge about literals. Second, the clause can be modified using a variable-value literal. Finally, the specialization can be performed by recursively repeating all steps for the current clause. At the end, redundant literals are removed from the clause. The program has a number of criteria that lead to improvements termination e.g. minimal number of examples that have to be covered by the clause, maximal number of literals in the clause, minimal improvement of new a clause in relation to its predecessor, etc. The target value for the regression is predicted based on *a regression plane through the class value of the covered examples* [62].

Other algorithm trying to deal with the regression problem is PCR [123, 124], which combines elements from two learning methods. The first one is unsupervised clustering while the second is supervised predictive modelling. In such combination the approach is called predictive clustering. The main idea of the PCR algorithm is to produce rules, which can be characterized as compact clusters of examples that have high similarity within the cluster, while they have high distance outside the cluster. Then the prediction of the target value is performed based on examples that belong to each cluster, e.g. using a simple average or a probability distribution across the discrete values.

Each cluster is represented in a rule form:

$$IF \;\; cluster\ description \;\; THEN \;\; target\ value$$

The rule induction process is based on the CN2 algorithm with modifications. The key difference lies in search heuristics that is used to guide the search for rules. The heuristics for CN2 is simply accuracy that focuses only on the target attribute. For the PCR algorithm this heuristics is inappropriate. The requirement of cluster compactness required to take into account not only the target attribute (which is common in predictive modelling) or conditional attributes (like in clustering) but all attributes. Therefore, an appropriate heuristics called dispersion is applied. The name of this heuristics has different meanings depending on the attribute type [123].

The dispersion for the nominal attributes is simply defined as normalized average Manhattan distance between the example and the frequencies vector of possible values within the set (called the prototype). At the end, the normalized distance is in the closed interval $[0, 1]$. For the numeric attributes the dispersion is presented as the variance:

$$s_N^2(E, a_j) = \frac{1}{N}\sum_{i=1}^{N}(x_{ji} - \bar{x}_j)^2$$

where E is an example set of size N, x_{ji} is the value of an attribute a_j, $\bar{x}_j$ is the mean of values of the attribute a_j. The normalization for numerical attributes is also performed but with the use of the standard deviation of the values of the attribute [123].

The second very important element distinguishing the PCR algorithm from CN2 is the treatment of examples covered by the rule. In the standard *covering* procedure such examples are removed from the training set. In PCR these examples are labelled with a lower weight. Thus, in the next iteration this example is less likely to be covered. Moreover if one example is covered a predefined number of times (in PCR this parameter is set to 5), then such an example is permanently removed.

The most interesting and particularly important are Janssen and Fürnkranz is works [57, 59]. In these works the authors describe the general idea of *the Separate-and-conquer* strategy for the regression rule induction and introduce the dynamic method of identification of positive and negative examples covered by the induced regression rule.

In the SeCoReg algorithm the main loop is identical as the one described in the previous work of Fürnkranz [34]. Briefly, the algorithm searches a rule that covers a part of examples yet uncovered by any of the rules. Then the covered examples are removed from the training set and the process is repeated until no instances remain.

However, the crucial part of the SeCoReg algorithm lies elsewhere.The heuristics used to control the process of induction is based on a novel approach called *dynamic reduction to classification*. In SeCoReg, each rule has a simple numerical value in its head part. This value is chosen as the median of the covered examples. The goal of the *dynamic reduction to classification* approach is to find covered examples that are close to the predicted value. In regression, however, the situation where the expected value is equal to the target value of all covered examples is rarely encountered. Thus, the natural consequence is to define the interval (error) in which examples can be found. For this purpose in SeCoReg the standard deviation is used. From a formal point of view, the example is labelled as positive if a distance between the target value and the predicted value is below the assumed threshold, otherwise it is labelled as negative [59]. It can be written in the form of:

$$class(e) = \begin{cases} positive & if\ |y_e - y_r| \leq t_r \\ negative & if\ |y_e - y_r| > t_r \end{cases} \tag{2.9}$$

where e is the example, y_e stands for the target value of the example, y_r is the predicted value for the rule, and t_r denotes the threshold.

The total number of positive and negative examples for the rule r are denoted then:

$$p_r = \sum_{i=1}^{k} (|y_i - y_r| \leq t_r); \qquad n_r = k - p_r$$

where k is the total number of examples covered by this rule.

The aforementioned formula is also used to define the total number of positive and negative examples for the training set. The main difference is that in that case the example may not be covered by the rule. The total number of positive and negative examples are denoted with capital P and N respectively. Then, k is the total number of examples in the training set.

It is worth mentioning that both formulas are marked with the r index. The reason for this marking is a dynamic change of parameters (p, n, P and N) for each candidate and/or rule in each refinement step. It means that a modification of a given rule may lead to different values of these parameters and thus to another quality assessment. Thanks to such a transformation from regression to classification problem, it is possible to apply classification quality measures for the assessment of regression rules.

There are also different approaches to deal with the regression problem that have nothing in common (in direct meaning) with *Divide-and-conquer* or *Separate-and-conquer*. The simplest approach to the induction of regression rules is discretization of the continuous decision attribute and the use of standard decision rule induction algorithms. Such an approach is presented by Torgo and Gama [120], who transform the continuous decision attribute into a set of intervals using three methods: equal-frequency, equal-width and k-means clustering.

In the equal-frequency intervals the algorithm divide instances into intervals with the same number of elements. In the equal-width method the range of values is divided by the number of clusters and then examples are assigned to one of new groups. However, in the k-means clustering method the clusters are created based on a function that minimizes the distance from the continuous decision attribute to the gravity center of interval [120]. The biggest problem of all three methods is the assumption that one knows the number of clusters.

By contrast to the simplest approach, the most computationally advanced methods of regression rule induction are based on ensemble techniques. The main aim of these algorithms is to increase the prediction performance based on linear combination of models instead of using simple models. Among others, one could mention RuleFit [30], its successor with some modifications FIRE [3] or RegENDER [24] that lead the rule induction towards minimization of the loss function calculated on the training or validation of a set of examples. To supervise the induction of subsequent rules, these algorithms apply various methods of optimization (e.g. gradient methods). The effects of their application are usually numerous sets of rules characterized by good quality of prediction.

The prediction is done by combining weighted voting. In the simplest form the main idea of the ensemble algorithms for prediction of the target value can be written as follows:

$$\hat{y} = f(x) = w_0 + \sum_{i=1}^{M} w_i \hat{y}_i$$

where w_0 is the baseline prediction and the sum part is treated as a correction of base value using the weighted value obtained from M rules [3]. In some ensemble algorithms there is no baseline prediction. In such cases the predicted value is calculated only from M models [17]. Although there are algorithms like e.g. FIRE where this equation is expanded by adding another sum, which implies further correction, the principal idea of using weights in all algorithms is common.

Nonetheless, all of ensemble techniques suffer from hindered interpretation of the results. Linear combination of decision trees, rules or other models is in fact much more complex than the model of a single tree or rule. Thus, clues, guides or additional methods are needed to improve the readability, transparency and comprehensibility of such models [17].

2.4. Rule and rule set quality

The accurate prediction of the target value is the key aim of all regression algorithms. There are many methods to create the single rule or the whole model in the form of a set of rules. Sometimes a new method is merely a modification of an existing one. Another time, the approach is completely innovative. In all cases, the quality criteria have to be established in order to assess the rule or the model.

In practice, rules and rule-based data models can be evaluated based on their own criteria e.g.: the most general model, the most interesting, the most useful for the user, the most unique etc. The criteria can be also different depending on a task or a problem domain. However, in order to evaluate in an objective manner which algorithm or method should be adopted to solve particular problem, one needs a systematic way to evaluate how good specific algorithms are in relation to the data set and to each other [110, 123, 127].

In the *Divide-and-conquer* and *Separate-and-conquer* strategies rules are assessed based on incomplete available information. In this situation the evaluation function is called *heuristic*. The main purposes of such functions is to make the best decision about the next step using partial information. The decision is, therefore, an approximate solution because it is made without information about the whole process. Heuristics are interchangeably called quality measures or simply evaluation metrics. As the primary goal of regression is accurate prediction, the heuristics are known here also as error measures.

The quality measures for regression can be generally divided into two groups. The measures from the first group operate on the principle of reducing the variance in the target value. This approach can be found in many works for both the *Divide-and-conquer* and *Separate-and-conquer* strategies, for example [57, 62, 120, 123, 127]. The appearance of these methods results from the impossibility to use quality metrics that measure simply the error rate. In regression the error cannot be measured as a correctly predicted value or not, but it is determined as the distance between the predicted and the real value.

Mean Absolute Error is a metric to obtain the averaged error between the predicted value $\hat{y}_i$ and the real value y_i without taking into account their signs.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{2.11}$$

Root Mean Squared Error is the most common measure used to calculate the error (not only in rule induction but also in other areas where the main aim is the prediction). This measure is also encountered without the root, however in the form of a square root the error is calculated on the same scale as the predicted value, which facilitates its interpretation. This measure has a tendency to emphasize the importance of outliers.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.12}$$

Both above mentioned measures for error prediction suffer from a common problem. These metrics refer to the absolute value of the error and it is meaningless to compare the averaged errors of the sets, where one can find the equally important error of 20% e.g. the error value 2 in a prediction of 10 and the error value 500 in a prediction 2500 in the second one. To avoid the problem of non-comparability, the normalization of errors should be done. The normalization is usually performed based on the total mean squared error of the default predicator [127].

$$MSE_{defualt} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{2.13}$$

where $\bar{y}$ is the mean value over the training data.

The measure that is the principal and comparable across different problem domains is **the relative root squared error** that in most cases is also multiplied by 100%:

Table 2.2: Contingency table with the established notations

		Predicted		
		Positive	Negative	
Actual	Positive	p (true positives)	P - p (false negatives)	P
	Negative	n (false positives)	N - n (true negatives)	N
		covered examples	not covered examples	P + N

$$RRSE = \frac{RMSE}{\sqrt{MSE_{defualt}}} \cdot 100\% \tag{2.14}$$

It is worth to note that this model compares the model to the simple predictor that is the average of the values over the training set. Moreover, it is important to emphasize that this measure can be used to evaluate the single rule, entire model or different regression rule learning algorithms. The value of the RRSE measure should be interpreted in relation to this average value. The smallest value is better while the value $> 100\%$ indicates that the model is a predicate, worst than the simple average.

On the contrary, the measures from the second group use the reduction to the classification approach which has been first used by Torgo and Gama [120]. The general idea of this method is to change the problem of regression into the classification problem (via discretization of the continuous decision attribute) and then use algorithms designed to solve the problem of classification (including the quality measures for classification). A slightly different idea has been presented by Janssen and Fürnkranz [59]. They proposed to use dynamic reduction to the classification approach for learning regression rules. This transformation is closely related to the induction process and allows to obtain statistics that are necessary to create a confusion matrix (also known as a contingency table) for each rule or refinement. Then, the matrix is used to estimate the rule quality.

The contingency table is a visualization of information about actual and predicted classification by a system. In other words the contingency table presents the classifier performance. The example of the contingency table is presented in Table 2.2.

Currently, in the literature one can find over 50 different quality measures to control the process of decision rule induction [15, 36, 58, 110]. The simplest statistics based on the contingency table determine the number of true positive (p) or false positive (n) examples, true positive rate ($\frac{p}{P}$) or false positive rate ($\frac{n}{N}$). However, in the classification approach the process of rule induction should be simultaneously optimized towards the two criteria: maximize the number of positive examples covered by the rule (also called *coverage* or *completeness*) and

Table 2.3: Definition of basic heuristics

name	heuristic formula
accuracy	p - n
precision	$\frac{p}{p+n}$
coverage	$\frac{p}{P}$

minimize the number of negative examples that are covered by the rule (*precision* or *sensitivity*) [4, 14, 15]. Therefore, these measures are not suitable as they optimize only one of two above mentioned criteria. The consequence of this is the appearance of quality metrics that take into account both criteria. Among them, the most common are precision, accuracy (p-n) or rule specificity and sensitivity $\frac{p}{P}$-$\frac{n}{N}$. Definitions of the most popular criteria in relation to the contingency table are presented in Table 2.3.

Moreover, analyzing results obtained by various rule induction algorithms one can safely state that description and prediction abilities of determined rules depend not only on the algorithm of searching for rules but also on a measure evaluating the quality of the induced rules. The quality measures applied in a rule induction algorithm are very important for final performance of an output rule set. This is confirmed by numerous empirical research works [4, 15, 16, 58, 102, 103, 110]. Sikora [110] noted that a few of quality measures lead to better results. These measures are:

C1 and C2 are quality measures proposed by Bruha in 1997 [14]. They are based on knowledge and observations of the author, who noted that two other quality measures which use statistics from the contingency table by *Coleman* and *Cohen* (also known as Cohen's Kappa coefficient) have deficiencies. The *Coleman* measure does not comprise the *coverage* while the *Cohen* measure leads to results that raise the importance of *coverage*. The numerical coefficients in both formulas stand for the normalization purpose [4, 14, 15, 102].

$$C1 = Coleman \cdot \left(\frac{2 + Cohen}{3}\right) \tag{2.15}$$

$$C2 = Coleman \cdot \left(\frac{P + p}{2P}\right) \tag{2.16}$$

where

$$Coleman = \frac{Np - Pn}{N(p + n)} \tag{2.17}$$

$$Cohen = \frac{(P+N)\left(\frac{p}{p+n}\right) - P}{\left(\frac{P+N}{2}\right)\left(\frac{p+n+P}{p+n}\right) - P} \tag{2.18}$$

Correlation (Corr) computes the correlation coefficient between the predicted and the target values. The Correlation measure is also used in the rule induction algorithm for subgroup discovery or association rule mining [58, 133].

$$Correlation\ (Corr) = \frac{pN - Pn}{\sqrt{(PN(p+n)(P-p+N-n)}} \tag{2.19}$$

g-measure (*g, g=2*) originally was proposed by Fürnkranz and Flach in 2005 [36]. It can be treated as a simple trade off between recall $\frac{p}{P}$ and precision $(\frac{p}{p+n})$ if $g = P$. However, a few authors [58, 110] have noted that the original version of this metric is too optimistic when the evaluation considers the rule that covers a small number of positive examples (e.g. rule that covers a positive example is characterized by precision equal 1). With modification the importance of such rules is decreased (precision of such a rule is equal to 0.33) but for a larger number of positive examples this correction has less and less influence.

$$g{-}measure\ (g,\ g{=}2) = \frac{p}{p+n+2} \tag{2.20}$$

s-Bayesian confirmation (*s*) has been proposed by Christensen [18] and Joyce [60]. In general, this measure presents an assessment of a degree in which a premise confirms a conclusion [16]. The first part of the measures evaluates the precision while the second is responsible for the reduction of the quality of the rule that covers a small number of examples [110].

$$s{-}Bayesian\ confirmation\ (s) = \frac{p}{p+n} - \frac{P-p}{P-p+N-n} \tag{2.21}$$

Logical sufficiency is a standard likelihood ratio statistics. The use of the logical sufficiency measure in the rule induction process leads to emphasize the precision of the rule at the expense of the number of covered examples [110].

$$Logical\ Sufficiency\ (LS) = \frac{pN}{nP} \tag{2.22}$$

Rule specificity and sensitivity is a measure that is approximately equal to Weighted Relative Accuracy (WRA) (The proof can be found in Fürnkranz and Flach is work [36]). It has been experimentally proved that the RSS measure leads to smaller rule sets than standard classification accuracy [119]. However, other research shows that RSS has a tendency to over-generalize [56].

$$Rule\ Specificity\ and\ Sensitivity\ (RSS) = \frac{p}{P} - \frac{n}{N} \tag{2.23}$$

Weighted Laplace is a modification of a standard Laplace measure. The task of both measures is very similar and is based on the estimation of the rule accuracy. With modification this measure takes into account the distribution of the number of positive and negative examples.

$$Weighted\ Laplace\ (wLap) = \frac{(p+1)(P+N)}{(p+n+2)P} \tag{2.24}$$

Regardless of the method of evaluation of a single rule or the entire set of rules there are also ways to assess theories. They are independent of the above heuristics and may be used both in the process of classification or regression problems. These measures could be also important and useful to meet their own criteria e.g. the most comprehensible model.

One of the most important measures of this group is the **size of theory**. It is simply defined as a number of rules ($\#rules$) contained in the theory. If multiple data sets are considered, this measure takes the form of an **average number of rules**:

$$average\ \#\ rules = \frac{1}{D}\sum_{i=1}^{D} R_i \tag{2.25}$$

where D is the number of data sets and R_i is $\#rules$ in the data set D_i.

Another interesting measure is the **number of conditions** in the rule set, but the **average number of conditions** in one rule seems to be more useful:

$$average\ \#\ conds = \frac{1}{R}\sum_{i=1}^{R} conds(R_i) \tag{2.26}$$

where R is the number of rules in a given data set and $conds(R_i)$ stands for the function that returns the number of conditions in rule i. For many data sets this value is averaged in the obvious way and described above.

There is also a measure of the **coverage of rules**, which presents the average number of examples from the data set that is covered by one rule.

$$cov = \frac{1}{R}\sum_{i=1}^{R} \frac{E_{R_i}}{E} \tag{2.27}$$

where R is the number of rules in the rule set, E_{R_i} is the number of examples covered by rule i and E is the number of examples in the data set. For multiple data sets the average coverage is calculated in a manner analogous to Equation 2.25.

It is noteworthy that the measures for the assessment of the theory in terms of its size are as important as the assessment of the accuracy of the model. The examination of the final data model can even be considered in the context of macro compromise between the size of the theory and its accuracy. In turn, the term of micro compromise should be understood as the induction of a single rule using a quality measure that optimizes the criteria of coverage and consistency simultaneously.

The problem of finding the trade off between the accuracy and size of the model is, however, still open. From one point of view the size of the theory is related to the principle of intelligibility proposed by Michalski and it would be best if the size of the theory was as small as possible, so that it would be easier to understand. Conversely, the accuracy of the model is usually better for more complex theories. Attempts to solve this problem are part of this dissertation. Thus, we will return to this discussion during the presentation of the experiments results.

2.5. Unordered rule set and resolving conflicts methods

The induced rule set, generally, may be either of the form of an unordered or ordered set of rules. In this work all presented rule induction algorithms rules are returned in the form of the unordered set. In contrast to the ordered rule set, where each example is covered by exactly one rule (the algorithm of prediction stops at the first rule that is satisfied), in the unordered set an example can be covered by several rules at the same time. The situation where two or more rules cover one example is commonly called the rules conflict and it leads to ambiguous estimation of the target value or classification in case of such a problem [21, 22, 33]. However, in real cases, the prediction for an unseen example should be clear and unambiguous, therefore a method for resolving conflicts is required with respect to the unordered set of rules.

It should be also noted that the problem of conflicts has not received very much attention in regression tasks. There are only few studies taking up research in this issue. In the case of classification problems the most popular solution is the so-called voting scheme in which a numeric value (that can be interpreted as the degree of confidence) is assigned to each rule in the conflict. Then the confidence degrees are summed up and the class with the maximum value of the sum is assigned to the unseen example [56, 76, 102, 110].

Interestingly, the algorithms that have been proposed to improve the accuracy of classification can also be treated as the methods of conflicts resolution. Examples of such algorithms are Double Induction and its next modification Recursive Induction [72, 74]. The main idea of the Double Induction algorithm is to repeat the rule induction based on the training examples that are covered by the rules in conflict. In Recursive Induction the process of Double Induction is just repeated if the new rule obtained with the use of Double Induction is still in conflict. The process of Recursive Induction continues until some stopping criteria are met.

Regardless of the number of iterations, both the Double Induction and Recursive Induction algorithms suffer from two limitations. The main limitation is the computational cost that is of course higher for the Recursive Induction algorithm. On the other hand, the authors have observed in experiments that the intersection of examples covered by the rules in the conflict is more important for new rules, which requires the selection of a certain weighting scheme for these examples. There is also a distance based method to solve the binary classification problem but the method can be used exclusively for the problem of numerical attributes [73].

In the absence of appropriate, fast and reliable methods that can be used to resolve conflicts in the regression rule induction algorithms in this work, the author proposed four complementary methods referred to as **mean of conclusions**, **mean of intersection**, **median of covered** and **max rule quality**.

The main advantages of all the presented methods is the speed of conflict resolution and obtaining the predicted value due to the simplicity of the algorithms. The first two methods (**mean of conclusions** and **median of covered**) are very obvious methods to check, however we would like to compare the simplest methods to the two original, more sophisticated and interesting approaches. These new approaches are **mean of intersection** and **max rule quality**.

1. **Mean of conclusions** - In the mean of conclusions method the expected value is predicted as the average value of all conclusions of conflicting rules. This seemingly obvious method, however, suffers from a common problem of all methods using mean value, namely vulnerability to statistical outliers. Another problem is the treatment of all rules equally, which is not appropriate when there is a large disproportion in the number of examples covered by the rules. As an example, let us assume that there are 2 rules in conflict. The first rule with target value x covers 500 examples while the second one with target value y covers only 5 examples. Thus, it is clear that the greater distance between x and y contributes to the greater prediction error.
2. **Median of covered** - The remedy for these problems is the second method, i.e median of covered. This method predicts the expected value as the median of the union of examples covered by conflicting rules. In principle, the median of covered method should better deal with outliers. It should also be more resistant to an uneven number of covered examples

by each conflicting rule, because the prediction will be based on the middle value of all examples without considering information about distribution.

3. **Mean of intersection** - The other approach to the outliers is presented in the method called mean of intersection. In the first step, the method takes into account only common examples for all conflicted rules. This is called the intersection. As a result, the outliers would not be taken into consideration, unless they are covered by all rules, which may mean that they are important. Then the method predicts the result as the average value of examples from the intersection.
4. **Max rule quality** - The last but not least method to resolve conflicts is the max rule quality method. In this method the prediction is performed using the heuristic that is also used in the induction process. The expected value is obtained from the conclusion of rule which has the highest value of quality measure used in induction. Other rules are then discarded. The main argument in support of the max rule quality method is a different approach to the estimation of the target value. Here, the prediction is not based on the target value of all examples that are covered by rules in conflict but on the most reliable subset of these examples, which is directly defined with respect to the quality measure and indirectly with the trade-off between two optimization criteria (*consistency* and *coverage*) of the rule.

2.6. Experimental evaluation of rule-based regression model

An appropriate approach to research is no less important than the choice of the algorithm or heuristic to control the process. In addition, the final model evaluation should be objective, repeatable and (in the case of writing one's own algorithm) comparable to existing methods. It is also worth remembering that it is essential that the results should be independent of the available data samples that are used to build the model. For example, one might encounter a situation where a data set is limited or a sample is not representative because of the uneven distribution of examples. In practice, the latter is ensured by the use of a cross-validation.

The cross-validation is a method for a reliable assessment of the prediction. The main idea is to split the data set into two subsets. The first one is used to train the model and therefore it is called a training set. The second one is applied for testing the model hence it is called a testing set. If learning schemas involve an additional stage to optimize some parameters after training or to evaluate a few created schemas on fresh data before the final error/accuracy evaluation, then the third subset, which is called validation set, is needed. If the optimization process is not required, then this set may be omitted. In this work, the validation set is not used, thus further considerations concern only the training and test sets.

The simplest way is to split the data set once and employ two-thirds of the data for training and the rest for testing. However, this process is heavily dependent on a single division (you can call it luck). Therefore, a different approach is utilized.

One of the most popular variations of this method is known as *k-fold cross-validation*. In this variant of cross-validation the data set is partitioned into k blocks (folds). Then each fold is successively used to test the model and the remaining $k-1$ folds are applied in the learning process. Usually, the process of creating folds should be carried out maintaining the distribution of the entire data set. In that case, the method is additionally described as the stratified k-fold cross-validation.

The process of separation can be illustrated as follows. The data set D is split into k blocks (U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9 and U_{10}). Then, in the first iteration the subset U_1 is taken for testing the model and the rest of folds (U_2,...,U_{10}) are used for training. In the second iteration U_2 is employed to test the model and U_1, U_3, ..., U_{10} are employed to build it. This process is repeated until each of the subsets will be used for testing. K obtained results are then averaged.

In the experimental evaluation of classification algorithms the most common is the 10-fold stratified cross-validation, however it is not proven that 10-folds is better than 5-folds, 20-folds or 1000-folds [127]. Due to the lack of defined classes in the regression problems the term stratification refers to the quantiles. In the case of 10-folds they are also known as deciles. Apart from the terms, the number of folds has significant meaning when it comes to consider the performance of algorithms. It seems that ten folds is an appropriate compromise between performance and independence of prediction.

Lastly, the cross-validation is sometimes repeated q times due to the variance reduction. Then formally the evaluation is performed based on the $q \times k$-fold cross-validation. The final performance estimation is made using the results from the k-fold cross-validation repeated q times. The q values are averaged again to obtain a single value. In the dissertation q takes the value 1, because of efficiency reasons. Nevertheless, the results are usually given for multiple data sets.

2.7. Statistical comparison of rule-based regression models

One of the final stages of the data mining analysis is a comparison of obtained results. The simplest approach is to analyze averaged values from different algorithms and different data sets. However, an additional element of this comparison is the use of statistical analysis, which allows you to specify whether there is enough evidence to say that one algorithm is better than another. In other words, the statistics *verify the hypothesis of improved performance* using tests [25].

In those tests the null hypothesis is that general performances of examined algorithms are equal. The hypothesis is then accepted (if there is no statistical significance between algorithms) or rejected (if one algorithm statistically differs from other algorithms) under a pre-defined significant level α (mostly 0.1, 0.05 or 0.01).

Statistical evaluation is also often treated as an essential part of the validation of machine learning algorithms [25]. The crucial work that unified an approach to evaluation of multiple algorithms on multiple data set is this of Demšar. Before this paper many researches have ended their work summarizing the results of statistical tests with the use of the matrix in which they compared all pairs of classifiers. In turn, the main aim of Demšar is work was to study, collect and systematize the statistical tests that could be used to compare two or more classifiers on multiple data sets.

To compare two methods or one method to the others, based on multiple data sets, Demšar has proposed to use the **Wilcoxon signed-ranks test** in place of the paired t-test, highlighting the weaknesses of this second one. In short, these are: an assumption of normal distributions due to the possession of small samples and certainty of commensurability over data sets. On the other hand, the Wilcoxon test is a non-parametric statistical test that does not assume normal distribution. Moreover, in the special case of the exponential or mixed-uniform distribution the Wilcoxon test is also more resistant to the outliers [25, 64]. The Wilcoxon test is only weaker in the case where assumptions of the paired t-test are met.

In the first step of the Wilcoxon test absolute differences between the performance of two classifiers are calculated. Then, these differences are ranked using these absolute differences and finally positive and negative ranks are added based on the formula:

$$R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i); \quad R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i);$$

where for differences = 0 ranks are split evenly for both sums with the assumption that the number of these differences is even, if not one of them is ignored.

In the last step, the test statistic T is calculated as a smaller value of the sums R ($T = min(R^-, R^+)$). The T statistic is then compared to the exact critical value for the Wilcoxon test. If the comparison takes into account about 25 sets [25, 135], methods or algorithms of the exact critical value can be read from one of many available statistical books. For a greater number of sets to test, the statistic T has approximately normal distribution (see Proof 2.7 for T^+. For T^- the proof can be performed in an analogous manner.) $N(\mu, \sigma)$ where

$$\mu = \frac{N(N+1)}{4} \qquad \sigma = \frac{N(N+1)(2N+1)}{24}$$

and the null-hypothesis can be tested using the Z-test

$$z = \frac{T-\mu}{\sqrt{\sigma}} = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \tag{2.28}$$

The *null hypothesis* with $\alpha = 0.05$ can be rejected if z is then smaller than -1.96.

Example. *Let $x_i = 1$ if the sign of the compared element is positive and 0 if it is negative. Considering the null-hypothesis, each of x_i has the Bernoulli distribution where $\mu = \frac{1}{2}$ and $\sigma = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$. Based on the Central Limit Theorem and using the properties of expected values we can then proof then*

$$\mu = E[T^+] = E[\sum_{i=1}^{n} ix_i] = \sum_{i=1}^{n} E[ix_i] = \sum_{i=1}^{n} iE[x_i] = \frac{1}{2}\sum_{i=1}^{n} i = \frac{1}{2}\frac{n(n+1)}{2} = \frac{n(n+1)}{4}$$

$$\sigma = var[T^+] = var[\sum_{i=1}^{n} ix_i] = \sum_{i=1}^{n} var[ix_i] = \sum_{i=1}^{n} i^2 var[x_i] = \frac{1}{4}\sum_{i=1}^{n} i^2 =$$
$$= \frac{1}{4}\frac{n(n+1)(2n+1)}{6} = \frac{n(n+1)(2n+1)}{24}$$

are exactly the same as μ and σ for normal distribution.

The testing procedure is the best to illustrate with an example. Suppose that Table 2.4 shows the comparison of average (over few data sets) performance (RRSE) of two algorithms. The experiment was performed on 8 quality measures and the null-hypothesis is that both algorithms performed equally well.

In Table 2.4 there is one quality measure which performed equally well in both algorithms A and B. The last column contains the rank assigned from the lowest to highest absolute difference. If two differences are equal then, the rank is averaged like for the LS and wLap measures where the following ranks should be 7 and 8 but both have 7.5. In the next step the ranks are summed up for positive and negative differences. The result is $R^+ = 29$ and $R^- = 1$. The critical value for the Wilcoxon test for a confidence level 0.05 and for 8 quality measures is 4. According to the definition of the test, if the smaller of sums from R^+ and R^- is equal or less than the obtained critical value, then the null-hypothesis is rejected. In the illustrated example the null-hypothesis is therefore rejected. Thus in this example it is credible to say that regardless of the chosen quality measure the A algorithm is better than the B algorithm.

Table 2.4: Performance comparison of two exemplary algorithms for carrying out the statistical evaluation of the significance using the Wilcoxon test

quality measure	algorithm A	algorithm B	difference	rank
C1	73.31	73.45	+0.14	4
C2	74.77	74.78	+0.01	2
Corr	77.65	77.71	+0.06	3
g-measure	88.61	88.61	0.00	1
s-Bayesian	82.15	82.41	+0.26	5
LS	72.53	72.88	+0.35	6.5
RSS	78.50	78.48	0.00	1
wLap	73.77	74.12	+0.35	6.5

For comparing more than two classifiers on multiple data sets Demšar has proposed to use the non-parametric **Friedman test** instead of repeated-measures ANOVA. The reasons for this recommendation are: an assumption of normal distribution in case of ANOVA and, what has been emphasized as more important, the assumption of sphericity (also known as circularity). This means that the variances of the differences between data sets are equal (e.g. between A and B, B and C and also A and C), not only with respect to the given samples but also to the population.

In contrast to the Wilcoxon test where ranks are determined for a single criterion, the Friedman test is based on the ranks of algorithms averaged over all tested criteria (e.g. quality measures as in previous example)

$$R_j = \frac{1}{N}\sum_{i=1}^{N} r_i$$

where r_i is a single rank calculated for i-th of N data sets for j-th of k algorithms. The ranks of algorithms performance ties of are averaged. Then the Friedman statistic is obtained using the formula

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{2.29}$$

with k-1 degress of freedom.

It should be borne in mind that χ_F^2 is distributed according to the F distribution for sufficiently large values of N and k, otherwise exact values have to be taken [25]. It has been shown [53], however, that the χ_F^2 statistics is too conservative, which can be understood as a lower ability to detect significant differences, and therefore a remedy has been proposed in the form of

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2} \tag{2.30}$$

In this test the null hypothesis H_0, which is that all classifiers are the same, is rejected under the predefined significant level, if the value of the F_F statistics is smaller than the critical value of the F distribution with $k-1$ and $(k-1)*(N-1)$ degrees of freedom. If the H_0 is rejected, the post-hoc **Nemenyi test** can be performed to compare algorithms to each other. In this test two compared classifiers are statistically different on the condition that the corresponding average ranks differ by at least a critical distance given by a formula

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{2.31}$$

where q_α is a critical value derived from the Studentized range statistics divided by $\sqrt{2}$. However, in case there are more algorithms in the comparison, the test of each pair of algorithms might be too time-consuming and require appropriate presentation of results. A good example of such presentation are critical difference diagrams [25], where statistical significant differences between multiple algorithms and over many data sets are drawn. The best presentation of the Friedman and Nemenyi tests, including critical difference diagrams, would be by an example.

Let us assume that the values of the error of 3 ($k = 3$) algorithms with respect to 12 ($N = 12$) data sets were obtained. The results are presented in Table 2.5. The rankings in parentheses are assigned in the order from the smallest to the largest value of the error for the given quality measure in each algorithm (by rows). In the last row the average rank of each algorithm is determined. The null hypothesis H_0 in this example is that all of three algorithms perform equally well.

$$\chi_F^2 = \frac{12 \cdot 12}{3 \cdot 4}\left[(1.917^2 + 2.792^2 + 1.291^2) - \frac{3 \cdot 3^2}{4}\right] = 13.64$$

$$F_F = \frac{11 \cdot 13.64}{12 \cdot 2 - 13.64} = 14.48$$

The critical value has to be obtained for $3-1=2$ and $(3-1)\cdot(12-1)=22$ degrees of freedom. At $\alpha = 0.05$, the critical value for $F(2,22)$ is 3.44. Therefore the null hypothesis is rejected. The Nemenyi test can thus be used to find difference between algorithms. For 3 algorithms the corresponding critical value at $\alpha = 0.05$ is 2.344 and the value of critical distance CD is $2.344 \cdot sqrt(\frac{3\cdot 4}{6\cdot 12}) = 0.9569$. Two algorithms then differ from each other if

Table 2.5: Performance comparison of three exemplary algorithms for carrying out the statistical evaluation of the significance using the Friedman test

data set	algorithm A	algorithm B	algorithm C
auto93	73.31 (2)	73.45 (3)	72.42 (1)
auto-mpg	74.77 (2)	74.78 (3)	72.99 (1)
auto-price	77.65 (1.5)	77.71 (3)	77.65 (1.5)
baseball	88.61 (2.5)	88.61 (2.5)	85.49 (1)
bodyfat	82.15 (1)	82.41 (3)	82.22 (2)
breasttumor	72.53 (1.5)	72.88 (3)	72.53 (1.5)
cholesterol	78.48 (2)	77.50 (3)	76.87 (1)
cloud	74.12 (2)	74.77 (3)	73.78 (1)
concrete	74.59 (1)	76.44 (3)	75.07 (2)
cpu	81.92 (3)	81.70 (2)	81.66 (1)
dee	73.21 (1.5)	74.22 (3)	73.21 (1.5)
diabates	76.13 (3)	76.09 (2)	75.86 (1)
average rank	1.917	2.792	1.291

the difference in the value of their ranks is greater than 0.9569. In the presented examples the differences between algorithms are: A to B 0.875, A to C 0.626 and B to C 1.501, but only for the last pair the C algorithm statistically outperforms the B algorithm.

In case they are more classifiers, the comparison can be performed in the same way, however, it is much more transparent when critical difference diagrams are used. Such a diagram, in relation to given average ranks, is presented in Figure 2.2.

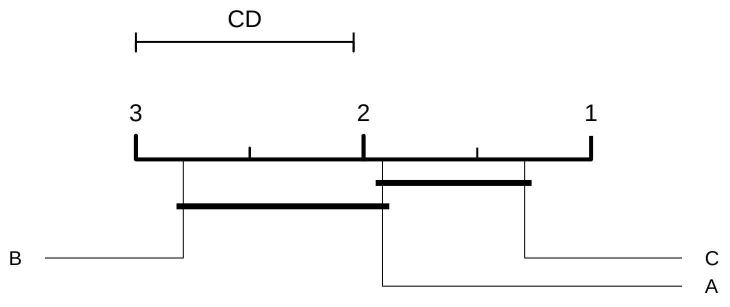


Figure 2.2: Comparison of all algorithms against each other with the Nemenyi test at $\alpha = 0.05$

Finally, it is worth mentioning that such non-parametric tests like the Wilcoxon test and the Friedman test can be used to compare many types of machine learning results including classification or prediction accuracies, error ratios, theory sizes, computation times or any other measures that can be compared with the use of ranks. This advantages and the previously mentioned ones have contributed to the high popularity of these tests [3, 7, 58, 86, 109, 110, 131].

3. Sequential rule induction algorithms

The existence of a number of classification rule induction algorithms is a fact. However, choosing the best one is not easy. Different algorithms can be characterized by different classification accuracy, the size of theories or by one's own subjective or objective criteria. Considering the problem from the other hand, the final evaluation (in fact the accuracy in classification) of the constructed model depends on the choice of the algorithm. In the case of regression rule induction the problem to choose the best or universal algorithm should be the same.

Please note that although different covering classification rule induction algorithms can be described by different performance or accuracy, they usually use one of two induction strategies: top-down or bottom-up. Moreover, these strategies describe the induction process in a general enough way that checking them both to tackle a regression problem seems to be a natural consequence. In this chapter, both approaches are presented taking into account the appropriate approach for regression.

First of all the *Top-down* strategy is presented. Later on, the precursory and unique approach to the rule induction using the *Bottom-up* strategy using centroids and k-d tree algorithm, is presented. The third part feature's a novel and innovative *Fixed* strategy for target value prediction. Finally, the author presents an unconventional approach to the knowledge discovery and the evaluation of regression rules with the use of confidence intervals.

3.1. Top-down strategy

The most common strategy for rule induction is a top-down one. The general idea is to define the most general problem and then divide it into smaller subproblems. This division is repeated for the subproblems until further division is impossible. In the end, the whole problem is presented by a set of smaller segments.

The approach is transferable to the regression rule induction algorithms. Let $ruleSet$ be a set of established rules which is initially an empty set (see the Algorithm 2, line: 2), $examples$ is a training set of examples and $ruleQualityMeasure$ is the selected heuristic to control the induction. The process of a single rule induction without any post-processing methods can be presented by only one phase where the rule grows (Algorithm 2, line: 5). In this phase the elementary conditions are successively added to the rule.

Algorithm 2 Pseudocode of the Top-Down regression rule induction algorithm

```
1:  function TopDownRuleInduction($examples, ruleQualityMeasure$)
2:      $ruleSet \leftarrow \emptyset$
3:      $uncoveredExamples \leftarrow examples$
4:      while $uncoveredExamples \neq \emptyset$ do
5:          $rule \leftarrow Grow(examples, uncoveredExamples, ruleQualityMeasure)$
6:          $covered \leftarrow Covered(rule, examples)$
7:          $uncoveredExamples \leftarrow uncoveredExamples \setminus covered$
8:          $ruleSet \leftarrow ruleSet \cup \{rule\}$
9:      end while
10:     return $ruleSet$
11: end function
```

The set of candidate elementary conditions is built as follows. For the numerical attribute the elementary condition takes the form of $a < q$ or $a \geqslant q$ where a is the name of the attribute under consideration and q is the average between two successive and unique values from the range of the sorted attribute a. For example, if the values 1.0, 3.0 and 6.0 belong to the attribute a then the conditions under consideration would be $a < 2.0$, $a \geqslant 2.0$, $a < 4.5$ and $a \geqslant 4.5$. If the attribute type is nominal, then the elementary condition takes the form of $a = q$ where q is a possible nominal value from the range of the attribute a.

The first set of candidate elementary conditions is obtained with the use of an entire set of training examples (Algorithm 3, line: 5). Further candidate sets are built based on a set of examples covered by the conjunction of condition which has been already added to the rule. Each candidate elementary condition is added to the temporary rule (Algorithm 3, line: 9) and such a rule is evaluated using the selected quality measure (heuristic) (Algorithm 3, line: 10). The condition for which the rule obtains the highest value of the heuristic is selected as the final one (Algorithm 3, line: 16). The growth stage ends when the rule does not cover any negative examples or when the addition of the next elementary condition does not change the set of examples covered by the rule (Algorithm 3, line: 4). The general outline of the grow procedure is presented by the Algorithm 3.

Another assumption that has been taken into account is that the missing attributes cannot be covered by any elementary condition. If the example has a missing or unknown value of any attribute, then the rule can cover this example only if the rule does not contain attributes whose values are unknown for this example. For example, if the rule contains two elementary conditions w_1 and w_2 for attributes named a and c and there would be an example with the

Algorithm 3 Pseudocode of the Grow procedure of the Top-Down algorithm

```
1: function GROW($examples, ruleQualityMeasure$)
2:     $rule \leftarrow \emptyset$
3:     $coveredExamples \leftarrow examples$
4:     while stop criterion do
5:         $conditions \leftarrow PossibleElementaryConditions(coveredExamples, rule)$
6:         $bestQuality \leftarrow -\infty$
7:         $bestCondition \leftarrow \emptyset$
8:         foreach $c$ in conditions do
9:             $temporaryRule \leftarrow addConditionToRule(rule, c)$
10:            $quality \leftarrow Evaluate(temporaryRule, examples, ruleQualityMeasure)$
11:            if $quality \geq bestQuality$ then
12:                $bestQuality \leftarrow quality$
13:                $bestCondition \leftarrow c$
14:            end if
15:        end foreach
16:        $rule \leftarrow addConditionToRule(rule, bestCondition)$
17:        $coveredExamples \leftarrow Covered(rule, examples)$
18:    end while
19:    return $rule$
20: end function
```

missing attribute b, then the rule can cover that example. But if the missing attribute would be a or c then the rule does not cover this example.

After that, the examples covered by the built rule (Algorithm 2, line: 6) are removed from the training set (Algorithm 2, line: 7) and the rule is added to the final set of rules (Algorithm 2, line: 8). The process is then repeated for remaining training examples (Algorithm 2, line: 4). The general outline of the algorithm can be represented by Algorithm 2.

3.2. Bottom-up strategy

The bottom-up strategy works in contrast to the top-down strategy direction. The main idea is to convert the problem presented in great detail to the problem described in a general fashion. The generalization process is repeated until the expected abstraction level is obtained. It is worth noting that in classification tasks it is a much less frequently used approach and mostly to the imbalanced data [86]. Moreover, in the regression tasks it is still an unexplored area of science.

With regard to the regression rule induction in this work, the bottom-up strategy is proposed as follows. Let $ruleSet$ is a set of established rules, $examples$ be a training set of examples and $ruleQualityMeasure$ is the selected heuristic to control the induction. The process of a single rule induction without post-processing can be divided into two phases: initialization (Algorithm 4, lines: $5-6$) and generalization (Algorithm 4, line: 7).

Algorithm 4 Pseudocode of the bottom-up regression rule induction algorithm

```
1: function BottomUpRuleInduction($examples, ruleQualityMeasure$)
2:     $ruleSet \leftarrow \emptyset$
3:     $uncoveredExamples \leftarrow examples$
4:     while $uncoveredExamples \neq \emptyset$ do
5:         $centroid \leftarrow DetermineCentroid(uncoveredExamples)$
6:         $rule \leftarrow CreateRuleForNearestExample(centroid)$
7:         $rule \leftarrow Generalize(rule, uncoveredExamples, examples, ruleQualityMeasure)$
8:         $covered \leftarrow Covered(rule, examples)$
9:         $uncoveredExamples \leftarrow uncoveredExamples \setminus covered$
10:        $ruleSet \leftarrow ruleSet \cup \{rule\}$
11:    end while
12:    return $ruleSet$
13: end function
```

In the initialization phase a certain rule must be specified. In order to choose this rule, the centroid method is proposed (Algorithm 4, line: 5). First, the centroid of all attributes is determined based on the training set. Then each example is taken under consideration to calculate the Euclidean distance from the centroid to the given example. The distance is measured by the formula

$$d = \sqrt{\sum_{i=1}^{k}(c_i - e_i)^2}$$

where i stands for the index of the attribute, c denotes the centroid and e stands for the example under consideration. In the case of symbolic attributes, the distance is fixed and takes the value of 0 if both values are equal, or 1 in other cases.

The example for which the value of the Euclidean distance is the smallest is selected. Then the most specific rule is built in such a way that the rule corresponds and finally covers the chosen example from the training set (this example can be called a *seed*) (Algorithm 4, line: 6). The body of the rule consists of conditions derived from the example. In the case of nominal attributes the elementary conditions take a form of ($a = q$) where q is a nominal value of attribute a from the example. On the other hand, for the numerical attributes the elementary conditions take a form of ($a < q$) and/or ($a \geqslant q$) where q is the arithmetic mean between two successive and unique values from the range of attribute a. At the end of this phase the rule has one or two elementary conditions for each numerical attribute, depending on the position of the considered value in the sorted set of all values of attribute a.

Table 3.1: Example of a data set used to induce regression rules using the bottom-up strategy

U	a	b	d
1	1.5	low	4
2	2	medium	3.5
3	3	medium	3
4	4	high	2

Two conditions ($<$ and $\geqslant$) are added to the rule in order to ensure that the built rule will cover only the *seed*. Therefore, one condition is added to the rule only for the minimum and maximum values from the range of numerical attribute a. Such a *seed* has only one neighbour, with a greater or lesser value depending on the used symbol comparison. Such a construction of rules can be illustrated by the examples from Table 3.1.

Let us assume that there is a data set U with four examples e_1, e_2, e_3 and e_4. Each example has only two conditional attributes. The first one holds numeric values and the second one nominal values. In order not to obscure the process, the numeric attribute a has already been sorted. The ordinal number is not involved in the induction process, however, has been retained to refer to specific examples.

The centroid in the first iteration for a given data set would be $a = 2.625$ which stands for the average value over attribute a and $b = medium$ that is a mode from an attribute b. The Euclidean distances are then $d_1 = 2.125$, $d_2 = 0.625$, $d_3 = 0.375$ and $d_4 = 2.375$ respectively for the corresponding examples. Thus, the initial rule comes from the example e_3. Then the initial rule would be as follows:

$$\boldsymbol{r_{init}} : \textit{IF}\ \ b = medium\ \ \textit{AND}\ \ a \geqslant 2.5\ \ \textit{AND}\ \ a < 3.5\ \ \textit{THEN}\ \ 3$$

where values for attribute a are calculated as the arithmetic mean between two successive values from the range a - i.e. 2.5 and 3.5.

In the generalization phase (see Algorithm 5), the rule built from the *seed* is iteratively subjected to generalization until an empty rule is obtained or a stop condition is reached. The process of rule generalization is done for each of k nearest examples determined by the k-dimensional tree algorithm (k-d tree) [29, 85], which can be understood as an extension of the nearest neighbour algorithm for binary space partitioning for k dimensions (Algorithm 5, line: 4).

Each iteration of generalization (Algorithm 6) boils down to the different examination of attributes depending on their type. In the case of nominal attributes, the condition is dropped if the rule and example have different values on it (Algorithm 6, lines: $5 - 6$). For numerical

Algorithm 5 Pseudocode of the Generalization procedure of the Bottom-Up algorithm

```
 1: function GENERALIZE($rule$, $uncoveredExamples$, $examples$, $ruleQualityMeasure$)
 2:     $possibleGeneralizations \leftarrow \emptyset$
 3:     while $stop\ criterion$ do
 4:         $neighbours \leftarrow GetNearestNeighbours(rule, uncoveredExamples)$
 5:         $bestQuality \leftarrow Evaluate(rule, examples, ruleQualityMeasure)$
 6:         $bestRuleGeneralization \leftarrow \emptyset$
 7:         foreach $n$ $in$ $neighbours$ do
 8:             $ruleGeneralization \leftarrow MostSpecificGeneralization(rule, n)$
 9:             $quality \leftarrow Evaluate(ruleGeneralization, examples, ruleQualityMeasure)$
10:             if $quality \geq bestQuality$ then
11:                 $bestQuality \leftarrow quality$
12:                 $bestRuleGeneralization \leftarrow ruleGeneralization$
13:             end if
14:         end foreach
15:         $rule \leftarrow bestRuleGeneralization$
16:         $allGeneralizations \leftarrow allGeneralizations \cup rule$
17:         $uncoveredExamples \leftarrow examples \backslash Covered(bestRuleGeneralization, examples)$
18:     end while
19:     $bestGeneralizedRule \leftarrow FindBestGeneralization(allGeneralizations)$
20:     return $bestGeneralizedRule$
21: end function
```

attributes (Algorithm 6, lines: $7-19$), the boundaries of intervals are extended to cover the example. The new boundary value is the arithmetic mean between the value of the example for attribute a and the successive value from the range of this attribute. These procedure is called *Most Specific Generalization* and its general outline is presented in Algorithm 6.

From all generalizations derived from the k nearest neighbours, the rule with the best value of the given heuristic is selected as the final one and the generalization process is repeated for this rule. When a stop condition is reached, the generalization phase ends and the rule with the highest value of the quality measure is picked (Algorithm 5, line: 19). The stop condition in the bottom-up strategy is the same as in the top-down approach where the rule has to cover at least a fixed number of examples. This also prevents from the generalization of an empty rule that would cover en entire data set of examples. To preserve the possibility of comparing algorithms, the value of a fixed number of examples has also been preserved.

Finally, the rule is added to the final set (Algorithm 4, line: 10). Then the examples covered by the built rule (Algorithm 4, line: 8) are removed from the training set (Algorithm 4, line: 9) and the process starting from determining the centroid is repeated for remaining training examples. In case the built rule covers an insufficient number of examples with regard to the assumed fixed number of examples, the *seed* example is removed from the training examples

Algorithm 6 Pseudocode of the Most Specific Generalization procedure

```
1: function MostSpecificGeneralization($Rule$, $Neighbour$)
2:     foreach $Attribute\ X_i$ do
3:         if $condition\ on\ X_i\ is\ missing\ in\ Rule$ then
4:             $Do\ nothing$
5:         else if $X_i\ is\ nominal\ and\ Neighbour_i \neq Rule_i$ then
6:             $Remove\ condition\ on\ X_i\ from\ Rule$
7:         else if $X_i\ is\ numeric\ and\ Neighbour_i > Rule_{i,upper}$ then
8:             if $Rule_{i,upper}\ is\ boundary\ value$ then
9:                 $Remove\ condition\ on\ X_i\ from\ Rule$
10:            else
11:                $Rule_{i,upper} = (Neighbour_i + Neighbour_{i+1})/2$
12:            end if
13:        else if $X_i\ is\ numeric\ and\ Neighbour_i < Rule_{i,lower}$ then
14:            if $Rule_{i,lower}\ is\ boundary\ value$ then
15:                $Remove\ condition\ on\ X_i\ from\ Rule$
16:            else
17:                $Rule_{i,lower} = (Neighbour_i + Neighbour_{i-1})/2$
18:            end if
19:        end if
20:    end foreach
21:    return $rule$
22: end function
```

and it is treated as uncovered. The process is then repeated for remaining examples. The general outline of the bottom-up algorithm can be represented by Algorithm 4.

3.3. Fixed strategy

The simple term of the numerical target value, in the context of regression, conceals a more complex issue. It was mentioned that in this work the rules are mostly induced with a single numerical value (the median of target values of examples covered by the rule) as a prediction but studies concern two different approaches. Taking into account the possibility of occurrence of outliers, we reject the averaged value of conclusion and in its place we study the median value of examples covered by the rule. In the second method the prediction of the target value is slightly different and its principle is derived from classification rule induction algorithms.

Algorithm 7 Fixed value modification (lines $3-4$) in the Grow procedure of Top-Down algorithm

```
1:  function GROW($examples, ruleQualityMeasure$)
2:      $rule \leftarrow \emptyset$
3:      $centroid \leftarrow DetermineCentroid(examples)$
4:      $rule \leftarrow AssignFixedTargetValueForNearestExample(rule, centroid)$
5:      $coveredExamples \leftarrow examples$
6:      while $stop\ criterion$ do
7:          $conditions \leftarrow PossibleElementaryConditions(coveredExamples, rule)$
8:          $bestQuality \leftarrow -\infty$
9:          $bestCondition \leftarrow \emptyset$
10:         foreach $c$ in $conditions$ do
11:             $quality \leftarrow Evaluate(rule \cup c, examples, ruleQualityMeasure)$
12:             if $quality \geq bestQuality$ then
13:                 $bestQuality \leftarrow quality$
14:                 $bestCondition \leftarrow c$
15:             end if
16:         end foreach
17:         $rule \leftarrow rule \cup c$
18:         $coveredExamples \leftarrow Covered(rule, examples)$
19:     end while
20:     return $rule$
21: end function
```

In the classification rule induction algorithms, the rule induction is performed for each decision class separately. Examples belonging to one of decision classes form a positive class while the remaining examples are included in the negative class. Then the rules are successively built until all examples from the positive class are covered by at least one rule. Then the next class is considered as positive and the process of single rule induction is repeated. The induction ends when all classes have been examined. Such an approach can be found in many classification rule induction algorithms [20, 110].

The key transferred idea that has been used in the proposed algorithm is the consideration of the classes in turn. The interpretation in regression rule induction algorithms puts each example and its target value down to the independent class. It means that the target value of a certain example (let us call it *seed* for both top-down and bottom-up strategy) is permanently assigned to the rule at the beginning and it does not change during the process of induction. It only remains to determine how this *seed* is chosen.

Algorithm 8 Fixed value modification (line 7) in the Bottom-Up algorithm

```
1: function BOTTOMUPRULEINDUCTION($examples, ruleQualityMeasure$)
2:     $ruleSet \leftarrow \emptyset$
3:     $uncoveredExamples \leftarrow examples$
4:     while $uncoveredExamples \neq \emptyset$ do
5:         $centroid \leftarrow DetermineCentroid(uncoveredExamples)$
6:         $rule \leftarrow CreateRuleForNearestExample(centroid)$
7:         $rule \leftarrow AssignFixedTargetValueForNearestExample(rule, centroid)$
8:         $rule \leftarrow Generalize(rule, uncoveredExamples, examples, ruleQualityMeasure)$
9:         $covered \leftarrow Covered(rule, examples)$
10:        $uncoveredExamples \leftarrow uncoveredExamples \setminus covered$
11:        $ruleSet \leftarrow ruleSet \cup \{rule\}$
12:    end while
13:    return $ruleSet$
14: end function
```

To permanently assign the target value to the rule we propose the method based on a centroid (Algorithm 7, line: 3 and Algorithm 8, line: 5). In the first step a middle point, which is called the centroid, of the uncovered examples is calculated. For attributes with numerical values we assume that the midpoint is the mean of all values of the given attribute, while for the nominal attributes the central point is a mode. After this assignment, the k-dimensional tree algorithm is used on the uncovered set of examples to find the example which has the smallest euclidean distance (the nearest example) to this centroid. Finally, the found example is marked as the *seed* and its target value is assigned to the empty rule (Algorithm 7, line: 4 and Algorithm 8, line: 7). The modifications for the Top-Down algorithm are outlined in Algorithm 7 while for the Bottom-Up algorithm in Algorithm 8. The process can be also illustrated by the example, however, it is very similar to the process of constructing the rule in the Bottom-Up algorithm as it was presented before for Table 3.1. In the Fixed value approach the assigned target value does not change during the process of induction.

Such an approach has no extra effects on the algorithm that uses the bottom-up approach. However, in the case of the top-down rule induction algorithm, the rule that is being built has to cover the *seed* too. Such an assumption is necessary because otherwise, the rule could cover examples completely unrelated to the estimated target value. And thus, this would have a huge impact on the prediction error. In the bottom-up approach this requirement is satisfied by the definition because the *seed* is always covered by the induced rule due to the specific induction process.

The effect of aforementioned assumptions is the study of four methods of prediction of the target value. These are:

1. The top-down algorithm with the median value as the prediction, which further in this work will be cited simply as **Top-Down algorithm (TD)**.
2. The top-down algorithm with the fixed value as the prediction, which further will be cited as **Top-Down Fixed algorithm (TDF)**
3. The bottom-up algorithm with the median value as the prediction, which further will be cited simply as **Bottom-Up algorithm (BU)**
4. The bottom-up algorithm with the fixed value as the prediction, which further will be cited as **Bottom-Up Fixed algorithm (BUF)**

3.4. Rule quality with a given confidence level

The quality of the rule, which is induced based on the data sample, is subjected to a distribution of this sample. In the perfect case the algorithm therefore should consider theentire population to prevent this problem. Considering the rule quality obtained on the basis of the sample (usually just the available set of data), the problem should be kept in mind. Alternatively, the rule can be evaluated in terms of statistics.

One of such approaches, where the generation of association and decision rules using, inter alia, confidence intervals, has been proposed by Wieczorek and Słowiński [126]. Due to the possibility of partial adaptation of the method in the induction of regression rules, the key used part of this approach is presented below.

Consider any rule in a form of $\varphi \to \psi$. In the method of dynamic reduction to classification, indirectly, one of the most important optimization parameters is ***p***, which stands for the number of examples satisfying the premise φ and the conclusion ψ and ***n*** which presents the number of examples satisfying the premise but not the conclusion. Assume further that the data set of examples (U, A) is a set of $|U|$ realizations of independent and uniformly distributed random variables. Moreover, each examples from U belongs to one of two aforementioned groups (***p*** or ***n***) or to the reminder group (not ***p*** and not ***n***). The probabilities of such an event can be written as (p_1, p_2, 1-p_1-p_2), where p_1 and p_2 stand for the probability of belonging to ***p*** or ***n*** respectively and

1-p_1-p_2 corresponds to a probability of another event. Events are also exhaustive and exclusive, which means that the occurrence of one event precludes the occurrence of another. Taking into consideration all the objects from the data set, one can therefore say that this data set is an example of multinomial random variables realization with unknown parameters (p_1, p_2, 1-p_1-p_2) and known parameters $|U|$. Thereby, the values p_1 and p_2 are true values of $\boldsymbol{p}$, which is often called support, and $\boldsymbol{n}$, which in turn is called anti-support of the considered rule obtained from a data set of a given multinomial distribution [126]. The estimation of these parameters can be performed using the total number of examples marked as positive and negative, respectively (see SeCoReg algorithm) divided by the size of the data set $|U|$

$$\hat{p} = \frac{p}{|U|} \qquad \hat{n} = \frac{n}{|U|} \tag{3.1}$$

The estimators of true values of parameters, calculated in this way, then can be used to determine *confidence intervals*. Typically, the confidence intervals are calculated for the confidence level α with one of the values: 1% or 5%. These confidence intervals for true values of p and n consist of pairs of values (p_{low},p_{high}) and (n_{low},n_{high}) which are simultaneously a lower and upper border of a given interval. Then the formal presentation of each interval is as follows

$$p_{low} \leq p \leq p_{high}$$

$$n_{low} \leq n \leq n_{high}$$

where the probability of the occurrence of true value within this interval is equal to $1 - \alpha$ [126].

The last step of definition of *confidence intervals* for p and n is the determination of values p_{low}, p_{high} and n_{low}, n_{high}. The authors [126] noted that the problem of the definition of the confidence interval had been taken into consideration by several researchers and allowed the development of a few types of methods, e.g. exact method, method based on bootstrap, method for small samples or method which uses approximation approach. The choice of defining the confidence intervals is not a part of this work, therefore we decided to use the method proposed by Gold [44]. The confidence intervals for p (but analogously for n) proposed by Gold have the form:

$$p_{low} = \hat{p} - \chi\sqrt{\frac{\hat{p}(1-\hat{p})}{|U|}} \tag{3.2}$$

$$p_{high} = \hat{p} + \chi\sqrt{\frac{\hat{p}(1-\hat{p})}{|U|}} \tag{3.3}$$

where χ is a square root of the critical value from chi-square distribution for one degree of freedom and two estimated parameters. In this work we assumed the confidence level $\alpha = 5\%$ to emphasize the proposed approach and to allow for a more visible presentation of the results. Then the resulting square root of the critical value is $\chi = \sqrt{5.02}$ for $\alpha = 0.05$ [126].

The defined confidence intervals lead to believe that the true value of the probability p and n lies somewhere within these intervals. From the rule quality point of view it is credible to say that the observations based on a data sample are then equally likely (with the probability 1-α) to the lower (p_{low}) and upper (p_{high}) solutions. In the best case the true values of $\hat{p}$ and $\hat{n}$ from Equation 3.1 can be then p_{high} and n_{low}. The quality of the rule with such values would be the highest possible one due to the biggest number of examples satisfying the premise and conclusion p and the lowest number of examples which satisfy only the premise n. Alternatively, the worst case and hence the worst rule quality would be with the opposite values. It means that p would be the lowest and n would be the highest. These approaches are called *optimistic* and *pessimistic* respectively.

In our algorithm we introduced the parameter for choosing the rule evaluation approach (*pessimistic*, *standard* or *optimistic*). The standard one is formulated as in the SeCoReg algorithm. The optimistic version is as follows. The estimation of parameters $\hat{p}_1$ and $\hat{p}_2$ is calculated based on the observed parameters p and n and the size of the data sample $|U|$. If $p_{1\ high}$ is denoted as $p_{highest}$ and $p_{2\ low}$ as n_{lowest} with respect to parameters p and n then

$$p_{highest} = \hat{p} + \sqrt{5.02}\sqrt{\frac{\hat{p}(1-\hat{p})}{|U|}}$$

$$n_{lowest} = \hat{n} - \sqrt{5.02}\sqrt{\frac{\hat{n}(1-\hat{n})}{|U|}}$$

Finally, the new values of p and n are assigned with values rounded to the nearest integer with the round half up tie-breaking rule

$$p_{new} = \left\lfloor p_{highest} \cdot |U| + \frac{1}{2} \right\rfloor$$

$$n_{new} = \left\lfloor n_{lowest} \cdot |U| + \frac{1}{2} \right\rfloor$$

It is noteworthy that in extreme cases the values of left and right endpoints are 0 and $|U|$ respectively. Considering the left case, the function that returns the number of covered examples cannot return the value less than 0. In the second case the rule quality is given with respect to the data sample, so in the best case the rule could cover at most all of the included examples.

The pessimistic version of the assessment of the rule quality is performed in a similar way with the assumption that the p_{lowest} and $n_{highest}$ values are determined. Apart from the chosen variant, the evaluation ends when all necessary statistics are obtained and the rule quality is calculated using one of the quality measures. This approach can also be understood as a change of value of the selected quality measure to get more pessimistic or more optimistic evaluation of the given rule.

4. Optimization of rules and rule sets

As it was mentioned before, the redundant rules can lead to the overfitting problem. This problem usually occurs when the model is defined too accurately, e.g. a solution is described with too many parameters thus it is more similar to the training examples rather than shows general trends or patterns occurring in them. As a result, the constructed model may perform well on the training data set but poorly on the unseen objects in the future [34, 69].

In order to avoid a situation in which the model becomes overtrained, the safety mechanisms are involved. Generally these mechanisms can be divided into two groups. In the first group, called *pre-pruning*, the optimization is performed during the induction process of each rule, therefore the algorithms which belong to this group are directly related to the specific rule induction algorithm. The main goal of these algorithms is to prevent overfitting of the rule to the training set of examples. This process can be also understood as a pruning phase of a single rule, because it follows right after the growth phase for each rule in the sequential covering rule induction process.

Conversely, the second group is known as *post-pruning* because the rule-based model is examined after the completion of its creation. The algorithms then involve the removal or transformation of one or more rules in such a way that the final rule set meets the selected optimization criteria, e.g. the error of prediction or the accuracy of classification are at least at the same level but the size of the theory is smaller. The *post-pruning* stage, which is also independent of the induction algorithms, is called *filtration*. Due to such independence, the focus will be put on rule filtering algorithms used in the stage of post-pruning, because these algorithms can be used to prune any unordered set of rules.

The research in the field of pruning algorithms mostly concerns classification systems but it has proven [32] that both approaches, *pre-* and *post-pruning*, have coped well with noise reduction independently. However, the *post-pruning* algorithms are inefficient, because they process a large amount of rules and their elementary conditions which are then pruned. As a remedy for the problem of wasting time, the combination of both phases has been proposed, e.g. in the TDP algorithm.

The numerous empirical studies show that in classification the number of rules after pruning and their classification abilities may vary depending on the *pre-* and *post-pruning* method [1, 32, 102, 103, 110, 131]. However, the regression rules are not as popular as their classification counterparts and the research in the area of regression relates rather to the new rule induction algorithms than to the filtration methods [24, 30, 59]. On the contrary, some classification rule filtering methods, like these based on genetic algorithms [1, 54], can be also applied to the regression rule sets. Nevertheless, the considerable group of filtration algorithms cannot be applied directly to the regression rule sets without any prior modifications.

The good optimization results obtained in the classification task, however, require verification in the regression problem. It is not clear whether the good results obtained in the classification guarantee similar results in the regression. Thus, this chapter presents two *pre-pruning* algorithms which are linked to rule induction algorithms presented earlier. Both *pre-pruning* algorithms are based on the hill-climbing approach. However, one of them proposes to use a more sophisticated approach using the Tabu list. In the following section 6 *post-pruning* algorithms are proposed. They could be used independently in any regression rule induction algorithms, because they do not interfere in the process of rule construction.

4.1. Algorithms for rule pruning

Pre-pruning algorithms are associated with already mentioned rule induction algorithms. Their main purpose is to prevent the occurrence of excessive overfitting rules to training data. For this purpose, the most frequently used mechanism is the one for the modification of the rule. In this work two such mechanisms are proposed. The first one uses a quality measure which has been selected for rule induction. In turn, the second one involves an additional mechanism to keep the best elementary conditions in the rule. An inspiration for this mechanism was a metaheuristic search method called *Tabu* that has been created by Fred W. Glover [41, 42].

It is also worth mentioning that the group of *pre-pruning* algorithm also includes heuristics, like the stopping criteria that are used to relax constraints of a perfectly fitted model. In classification, the most popular relaxation method is to allow the rule to cover some of negative examples while some of positive examples could be still unexplained [21, 31, 32, 92]. In the case of the regression problem, this approach can be utilized as well, together with the approach of dynamic reduction to classification [59]. Therefore, in our implementation we are using such a method too. A bigger number of positive examples and smaller number of negative are obviously better but we also accept a partial coverage and negative examples.

In addition, in order to prevent the induction of too specific rules, the candidate condition in the Top-Down strategy is considered only if the rule with such a condition covers at least a fixed number of examples which are not covered by rules generated so far. On the contrary, in the Bottom-Up induction strategy the rule after generalization also has to cover at least a fixed number of examples.

Apart from the induction strategy, a similar overfitting mechanism is often incorporated in various implementations of a decision tree and rule learners in the form of the parameter specifying the minimum number of examples per leaf/rule [127]. Naturally, the value of this parameter often requires to adjust to a particular dataset, however in this work, where experiments are performed for few algorithms and multiple datasets, the value of this parameter is set to 3, which seems to be a reasonable minimum for the calculation of the median and the standard deviation of the rule conclusion.

In addition to stopping criteria, we propose pruning mechanisms called *Heuristic pruning* and *Tabu pruning*. Both algorithms try to produce a shorter version of the rule without sacrificing its quality. In both algorithms the main evaluation criterion is the selected quality measure which allows to assess the rule before and after the application of *pre-pruning* algorithm. In the proposed implementation we assumed that the rule quality after pruning cannot decrease below the value before. Thus, pruning is continued until the removal of the elementary condition improves the rule quality or at least keeps it at the level before deletion. A slightly modified assumption has been applied in the paper [111], where the authors have proposed a hierarchical strategy to improve classification accuracy with the combination of the original rule and shortened rules whose quality may be 5 to 40% worse than the original rule. Since the proposed algorithms modify the original rule, we assume that the quality of this rule cannot be lower.

4.1.1. Hill climbing pruning

The simplest method to prune a rule from redundant elementary conditions involves the sequential removal of these conditions, which most improves or at least does not decrease the quality of the rule with respect to the chosen quality measure. In the beginning all conditions have to be checked (Algorithm 9, line 4). For this purposes, each condition is temporarily removed from the rule and the quality of such rule is evaluated (Algorithm 9, lines: $5-6$). If the quality of the rule without the condition is better than the previously achieved best value, then the new value is stored and it is treated as a new reference value (Algorithm 9, line: 11). Regardless of the result the condition returns to the rule and the process is repeated for remaining conditions (Algorithm 9, line: 7). Finally, the the elementary conditions for which the rule has obtained the greatest improvement in the quality are deleted (Algorithm 9, line: 17). The removal process is then repeated as long as the quality of the rule is not degraded and the number of conditions is bigger than 1. The outline of this *hill climbing pruning* algorithm is depicted in Algorithm 9.

Algorithm 9 Hill climbing pruning algorithm

```
1: function PRUNERULE($rule$, $examples$, $ruleQualityMeasure$)
2:     $bestQuality \leftarrow Evaluate(rule, examples, ruleQualityMeasure)$
3:     $worstCondition \leftarrow \emptyset$
4:     $conditions \leftarrow getConditions(rule)$
5:     repeat
6:         $quality \leftarrow \emptyset$
7:         foreach $condition\,in\,conditions$ do
8:             $removeCondition(rule, condition)$
9:             $quality \leftarrow Evaluate(rule, examples, ruleQualityMeasure)$
10:            if $quality \geq bestQuality$ then
11:                $bestQuality \leftarrow quality$
12:                $worstCondition \leftarrow condition$
13:            end if
14:            $restoreCondition(rule, condition)$
15:        end foreach
16:        if $worstCondition$ then
17:            $removeCondition(rule, worstCondition)$
18:        end if
19:    until $worstCondition \neq \emptyset$
20:    return $rule$
21: end function
```

4.1.2. Tabu hill climbing pruning

To treat all elementary conditions equally is the simplest approach. This strategy, on the other hand, does not protect elementary conditions which have a huge impact on the exacerbation of the rule quality at the beginning of the pruning process. In subsequent iterations of the algorithm, these elementary conditions are treated in the same way again without consideration of the reverse memory. As a result, such an assumption can cause some side effects. Firstly, this is a perfect example of local search methods which generally have a tendency to stack in a suboptimal solution. Secondly, it may cause the removal of the condition that reduces a descriptive ability, because it takes into account only the prediction ability of the rule.

In a view of the above problems we propose to modify the *Hill climbing pruning* algorithm, which uses a memory structure to remember the essential conditions and protect them from re-examining and removal in the future. Our inspiration here is a method already known as *tabu search*. It has been proposed by Fred W. Glover [41, 42] as an algorithm for solving combinatorial optimization problems and in the current implementation it is used to solve many problems in the fields of planning resources, financial analysis, scheduling, energy distribution, biomedical analysis, pattern classification and many more [43, 114].

The main idea of the algorithm, which we called *Tabu hill climbing pruning*, is to remember one elementary condition (in each iteration) carrying the most information, which is understood here as the biggest influence on the deterioration of the rule quality. For this purpose, we used the mentioned memory structure.

Algorithm 10 Tabu hill climbing pruning algorithm

```
1:  function PRUNERULE($rule, examples, ruleQualityMeasure$)
2:      $bestQuality \leftarrow tabuQuality \leftarrow Evaluate(rule, examples, ruleQualityMeasure)$
3:      $tabuList \leftarrow \emptyset$
4:      repeat
5:          $quality \leftarrow worstCondition \leftarrow \emptyset$
6:          foreach $condition\ C_i$ do
7:              if $tabuList\ does\ not\ contain\ C_i$ then
8:                  $quality \leftarrow Evaluate(rule \backslash C_i, examples, ruleQualityMeasure)$
9:                  if $quality < tabuQuality$ then % this is
10:                     $tabuQuality \leftarrow quality$ % unique part
11:                     $bestCondition \leftarrow C_i$ % of Tabu algorithm
12:                 end if
13:                 if $quality \geq bestQuality$ then
14:                     $Remember\ new\ best\ quality\ and\ the\ worst\ C_i$
15:                 end if
16:                 $restoreCondition(rule, C_i)$
17:             end if
18:         end foreach
19:         if $bestCondition$ then % only for
20:             $addCondition(tabuList, bestCondition)$ % Tabu algorithm
21:         end if
22:         if $worstCondition$ then
23:             $removeCondition(rule, worstCondition)$
24:         end if
25:     until $worstCondition \neq \emptyset$
26:     return $rule$
27: end function
```

In each iteration one elementary condition is checked twice (Algorithm 9, lines: $9 - 12$ and $13 - 15$). One assessment is performed in the same way as in the previous algorithm (to check whether the removal of the elementary condition has a positive impact on the rule quality or not). Conversely, there is additional *if statement* checking if the removal of the same condition has negative influence on the rule. It is worth mentioning that expressions are separated. This is because the quality of *tabuQuality* refers to the quality of the worst condition while *bestQuality* stands for the value of the best condition and both elementary conditions are not linked. Moreover, the value of the elementary condition, which is not better than the value of *bestQuality*, is not clearly worse than *tabuQuality*. After all the conditions are checked, the best one is added to *tabuList* (Algorithm 9, line: 20) and the worst one is permanently removed from the rule (Algorithm 9, line: 23). The process is then repeated for remaining conditions. The sketch of the *Tabu hill climbing pruning* algorithm is presented in Algorithm 10.

4.2. Algorithms for rule filtering

The readability of a rule-based model can be decreased by the occurrence of rules which are redundant within the meaning of redundancy as a certain criterion, chosen by the user of the system, of the uselessness of rules. In this work one can distinguish two such criteria. On one hand the rule is redundant if it covers examples that are also covered by other rules in the rule set. This is because the sequential covering induction strategy does not prevent the occurrence of such rules [110]. Thus, if there are rules that cover a subset of examples already covered by other rules, then those rules theoretically contribute nothing and only blur the structure and readability of the model.

From the other point of view, the redundancy of rules can be examined in the context of predictive abilities. In such context the rule is redundant if it does not have a positive impact on the predictive ability of the whole rule set. In this case the rules may have low coverage, but what is the most important, their conclusion is far from the true value and the mere presence of such a rule in the rule set only spoils the quality of the classifier.

The simplest idea to cope with the aforementioned redundant rules is to remove them. However, the problem of finding a minimal subset of the set of rules, which satisfies the established criteria (e.g. overall classification accuracy, balanced accuracy, relative mean squared error), is NP-complete and computationally expensive [1, 5], and hence it is not likely to be efficient to find out the solution in a direct way. Consequently, to deal with this problem the heuristic search algorithms are used during the filtration.

One of such heuristic methods for searching a quasi-minimal subset of the set of rules has been proposed by Ågotnes et al. In this work the authors, proposed a genetic algorithm. In this genetic algorithm an individual is specified as a classifier and the occurrence of the rule in the classifiers is expressed by the value of 1 on an appropriate position of the encoded individual. The optimization is then performed toward maximization of a function that is a weighted sum of the overall classification accuracy and the inverse of the model complexity (e.g. the number of rules). For the filtration of fuzzy rule sets, the method of the genetic algorithms family has been also presented by Ishibuchi et al [54].

With respect to the presented work, the most important are the *Inclusion*, *Coverage*, *Backward* and *Forward* algorithms proposed by Sikora [102, 103] and Wróbel [110] for filtration of classification rules. However, it seems that the application of all these algorithms to solve the regression problem is also feasible and, what is more important, can bring a satisfactory outcome. Thus, an important contribution of this work was to adapt the aforementioned algorithms for regression. It means that the algorithms were designed to evaluate the regression rules using any of the presented quality measures and to evaluate the rule set using RRSE. The results of this research have been already published in our two papers [109, 131].

Studies on the quality-based filtration have been also conducted by Øhrn et al [88] and Ågotnes et al [1]. The main objective of their research was to obtain a certain number of the best rules according to the selected quality measures and then evaluate again the reduced rule-based model. By plotting the number of the top rules versus the performance of the model it is possible to select an appropriate number of rules according to the specific problem.

Research on the filtering algorithm that eliminates the rules whicg cover a similar set of examples was conducted by Gamberger and Lavrač [39]. The characteristic idea of their work was the assumption that rule does not necessarily have to come from one induction algorithm. Moreover, rules can be even defined by the expert and then joined with the rules obtained by automatic induction algorithms. Thus a prepared set of rules is then subjected to the filtration algorithms, which eliminates redundant rules, starting from the rules that cover the smallest number of positive examples.

In systems where the classification ability is not a crucial criterion, for example in the case of the rules for descriptive purposes, the filtration can be performed based on the minimum quality requirements. Generally, such a requirement is defined as a measure of the rule attractiveness, e.g. a statistical significance [50, 121]. There are also a number of complex measures that allow to combine multiple quality requirements and obtain a set of rules with respect to the value of these measures [105]. As an example of such a complex measure one can give the sum or the weighted multiplication of all quality requirements or the specific lexicographical order (e.g. more important rules are checked first and then the less important ones). For certain rule interestingness measures Bayardo and Agrawal have proved that the best rules according to these measures reside on the Pareto-optimal border with respect to support and confidence [8]. Brzezińska, Greco and Słowiński attempted to continue these studies and proposed a new evaluation space formed by the rule support and anti-support. The authors have also proved that this space is the upper set of the support-confidence set and contains all rules optimal with respect to some attractiveness measures [16].

In this work we propose six algorithms: *Inclusion*, *Coverage*, *Disjoint*, *Forward*, *Backward* and *ForwBack* for filtration of regression rules. Bearing in mind the criteria, which have been mentioned in the first paragraph of this section, one can divide these algorithms into two groups. The first group consists of the algorithms that focus on the optimization towards the best prediction accuracy. This group may include algorithms: *Forward*, *Backward* and *ForwBack*. On the contrary, the second group consists of algorithms with the task of filtering the rules, which cover a similar subset of examples. To this groupthe following algorithms belong: *Inclusion*, *Coverage* and *Disjoint*.

All presented algorithms have very similar input parameters. For each algorithm there should be a pass: the set of rules, the verification set of examples and the selected quality measure. Another parameters are directly related to some algorithms and they will be discussed later. The output of the algorithms is a reduced rule-based model.

4.2.1. Inclusion

The general outline of the *Inclusion* algorithm is presented in Algorithm 11. The main idea of the *Inclusion* algorithm is to remove rules if the rule set contains another rule, which is supported by the same examples and the quality of the other rule is higher than the quality of the tested rule (Algorithm 11, line: 11). The example supports the rule if it satisfies the premise and if its target value is in the range of $v \pm sd$, where v is the median value of examples covered by the rule and sd is the standard deviation from the target value of these examples. The examples supporting the rule are returned by the function called *Supported* (Algorithm 11, line: 10). In this algorithm, there is also one auxiliary function *WorstRule* (Algorithm 11, line: 8). The main task of this function is to return the rule characterized by the worst quality with respect to the chosen measure and a given dataset.

Algorithm 11 *Inclusion* rule filtering algorithm

```
1: Input:
2: rules: a set of rules
3: dataset: a training set of examples
4: q: rule quality measure
5: Output: a filtered set of rules
6: outputRules ← $\emptyset$
7: while (rules $\neq \emptyset$) do
8:     rule ← WorstRule(rules, dataset, q)
9:     rules ← rules$\setminus\{$rule$\}$
10:    supp ← Supported(rule, dataset)
11:    if (foreach r in rules supp$\setminus$Supported(r, dataset) $\neq \emptyset$) then
12:        outputRules ← outputRules $\cup$ $\{$rule$\}$
13:    end if
14: end while
15: return outputRules
```

4.2.2. Coverage

The general outline of the second *post-pruning* method called *Coverage* algorithm is presented in Algorithm 12. The *Coverage* algorithm is very similar to the *Inclusion* one. The main idea in both algorithms is almost identical, except that the *Coverage* algorithm removes the rules that support examples that have been already supported by the previously examined rules (Algorithm 12, line: 12). It means that if the considered rule covers at least one, uncovered so far, example from the training data set, then the rule carries important and unique information and has to be added to the final set.

In addition, this rule also meets the assumption that the previously examined rules have a higher quality due to the inverse effect of additional function (*BestRule* - Algorithm 12 line: 9), which in this case returns the best rule according to the selected quality measure and to the used dataset of examples.

Algorithm 12 *Coverage* rule filtering algorithm

```
 1: Input:
 2: rules: a set of rules
 3: dataset: a training set of examples
 4: q: rule quality measure
 5: Output: a filtered set of rules
 6: outputRules ← $\emptyset$
 7: suppSum ← $\emptyset$
 8: while (rules $\neq \emptyset$) do
 9:     rule ← BestRule(rules, dataset, q)
10:     rules ← rules$\setminus\{$rule$\}$
11:     supp ← Supported(rule, dataset)
12:     if (supp$\setminus$suppSum $\neq \emptyset$) then
13:         outputRules ← outputRules $\cup$ $\{$rule$\}$
14:         suppSum ← suppSum $\cup$ supp
15:     end if
16: end while
17: return outputRules
```

4.2.3. Disjoint

The *Disjoint* algorithm is the third algorithm from the group of algorithms based on the coverage criteria. However, its aim and principle of operation are quite different than of the two previous algorithms. The *Disjoint* algorithm tries to find rules that describe the set of examples by the most disjoint groups and, at the same time, it should produce the smallest possible set of rules.

The *Disjoint* algorithm has one extra parameter (with respect to *Inclusion* and *Coverage* algorithms) $minCov$, which is the minimum satisfied coverage of the set of rules after filtration. This parameter takes continuous values in the range 0-1, where 0 stands for the lack of coverage and 1 denotes the full coverage of training examples. The value of $minCov$ is taken into account inside an outer loop of the algorithm (Algorithm 13, line: 10) which allows to obtain the established coverage by the set of filtered rules (in the extreme case this coverage is gained by the first best rule). In contrast, the inner loop of the algorithm (Algorithm 13, lines: $12-18$) allows to obtain a rule that gives the highest coverage. Such a best rule is finally added to the rule set (Algorithm 13, line: 19) and the process is repeated for the rest of the rules until the minimum satisfied coverage is not met. The algorithm uses one auxiliary function called *Card* which returns the cardinality of the given set of examples. The sketch of the *Disjoint* algorithm is presented in Algorithm 13.

Algorithm 13 *Disjoint* rule filtering algorithm

```
1: Input:
2: rules: a set of rules
3: dataset: a training set of examples
4: minCov: minimum required coverage of dataset
5: q: rule quality measure
6: Output: a filtered set of rules
7: bestRule ← BestRule(rules, dataset, q)
8: outputRules ← {bestRule}
9: suppSum ← Supported(bestRule, dataset)
10: while ($\frac{\mathrm{Card(suppSum)}}{\mathrm{Card(dataset)}}$ < minCov) do
11:     n ← 0
12:     foreach (r in rules) do
13:         supp ← Supported(r, dataset)
14:         if (Card(supp\suppSum) > n) then
15:             n ← Card(supp\suppSum)
16:             bestRule ← r
17:         end if
18:     end foreach
19:     outputRules ← outputRules ∪ {bestRule}
20:     rules ← rules\{bestRule}
21:     suppSum ← suppSum ∪ Supported(bestRule, dataset)
22: end while
23: return outputRules
```

4.2.4. Forward

The *Forward* algorithm is the first algorithm from the group of algorithms that are not based on the information about rules coverage. The key idea of the *Forward* algorithm is to optimize the given set of rules towards certain evaluation criterion, which is also given as an additional parameter of this algorithm. In our implementation the evaluation criterion of the rule set is the same at each stage and it is a *RRSE*, but it could be any other objective or subjective criterion to assess the set of regression rules (e.g. correlation coefficient, relative-absolute error, mean-absolute error, mean-squared error etc.).

The *Forward* algorithm starts from the best rule obtained by the *BestRule* method (Algorithm 14, line: 10), which returns the rule with the highest value of the specified quality measure. In the next step the best rule is added to the new empty rule set (Algorithm 14, line: 12). In the following iterations the algorithms check if the addition of the next rule reduces the RRSE of the rule-based model (Algorithm 14, lines: $12 - 14$). If so, then this rule is added to the final rule set, otherwise the rule is removed. The process continues until the rules remain in the input rule set (Algorithm 14, line: 9). In the end, the filtered rule set is returned.

Algorithm 14 *Forward* rule filtering algorithm

```
1: Input:
2: rules: a set of rules
3: dataset: a training set of examples
4: criterion: evaluation criterion
5: q: rule quality measure
6: Output: a filtered set of rules
7: outputRules ← $\emptyset$
8: e ← Evaluate(outputRules, dataset, criterion)
9: while (rules $\neq \emptyset$) do
10:     rule ← BestRule(rules, dataset, q)
11:     rules ← rules$\setminus\{$rule$\}$
12:     outputRules ← outputRules $\cup$ $\{$rule$\}$
13:     e’ ← Evaluate(outputRules, dataset, criterion)
14:     if (e’ is better than e) then
15:         e ← e’
16:     else
17:         outputRules ← outputRules$\setminus\{$rule$\}$
18:     end if
19: end while
20: return outputRules
```

4.2.5. Backward

The *Backward* algorithm works in contrast to the *Forward* algorithm but with the same number and types of parameters. The rule filtering process starts from invocation of the *WorstRule* method, which returns the worst rule according to the chosen rule quality measure. Then the rule is temporarily removed from the entered rule set (Algorithm 15, line: 12) and the evaluation of the smaller set is performed (Algorithm 15, line: 13). Then the quality values of the smaller and bigger rule sets are compared and if the smaller rule set has better quality, the rule is permanently removed (Algorithm 15, line: 14). Otherwise, the rule returns to the rule set (Algorithm 15, line: 17). The process is maintained until all rules are checked. The pseudo code of the *Backward* algorithm is shown in Algorithm 15.

Algorithm 15 *Backward* rule filtering algorithm

```
1: Input:
2: rules: a set of rules
3: dataset: a training set of examples
4: criterion: evaluation criterion
5: q: rule quality measure
6: Output: a filtered set of rules
7: outputRules $\leftarrow$ rules
8: e $\leftarrow$ Evaluate(outputRules, dataset, criterion)
9: while (rules $\neq \emptyset$) do
10:     rule $\leftarrow$ WorstRule(rules, dataset, q)
11:     rules $\leftarrow$ rules$\setminus\{$rule$\}$
12:     outputRules $\leftarrow$ outputRules$\setminus\{$rule$\}$
13:     e' $\leftarrow$ Evaluate(outputRules, dataset, criterion)
14:     if (e' is better than e) then
15:         e $\leftarrow$ e'
16:     else
17:         outputRules $\leftarrow$ outputRules $\cup$ $\{$rule$\}$
18:     end if
19: end while
20: return outputRules
```

4.2.6. ForwBack

The last algorithm from the group of algorithms which focus on optimization of the rule set towards the lowest prediction error is the *ForwBack* algorithm. It works like a combination of the two above mentioned *Forward* and *Backward* algorithms. First, the given rule set is entered as a parameter to the *Forward* algorithm and then the result of this algorithm is passed as a parameter (a set of rules) to the *Backward* algorithm. The main objective of this approach is to identify the rules whose quality is not the worst or the best (so the rule quality is located somewhere in the middle of the ranking list), but their removal has a positive impact on reducing the prediction error.

5. Experiments

In this chapter the experimental evaluation of all theses, which have been previously discussed, is performed. First, the test domains and evaluation methodology are presented. Then each experiment is introduced via a common schema. The experiment begins with a short resembling introduction to the problem. Then the key and studied part of experiment are defined. Finally the results are shown and discussed.

It is worth noting that the presented experiments could be divided into two groups. The first group presents the results of experiments for each component separately. This means that a comparison of several of the proposed approaches will be conducted with respect to each other. In turn, the second group of experiments relates to the comparison of the algorithms constructed from the best components in relation to the results from some existing learning algorithms. At the end of this chapter the summary of the both groups of experiments is presented.

5.1. Test domains

Experimental evaluation of the proposed approaches was carried out on 30 publicly available (e.g. in UCI Machine Learning Repository [6]) benchmark data sets listed in Table 5.1. The data sets were selected to cover a wide and different range of various data. Thus, the data sets differ in the number of examples and attributes, as well as in the number of distinct values of the target attribute or in the scale of these values.

5.2. Default settings in algorithms

The list of default parameters of the program is presented in Table 5.2. We have made every effort to ensure that the number of parameters of the algorithms was as small as possible. Some assumptions, however, are necessary. One of such assumptions is the minimal number of covered examples to deal with over the fitting problem. To read more about this parameter see Section 4.1. The second fixed parameter is the maximal number of neighbours that are taken into consideration for generalization of the rule in the case of bottom-up algorithms. Other parameters that can be also treated as variables of algorithms are covered by at least one experiment.

Table 5.1: Regression data sets and their characteristics: number of examples, number of all attributes, number of nominal attributes, number of numeric attributes, and number of distinct values of the target attribute.

data set	#examples	#attributes	#nominal attr	#numeric attr	#distinct values
auto93	93	22	6	16	81
auto-mpg	398	7	3	4	129
auto-price	159	15	1	14	145
baseball	337	16	0	16	208
bodyfat	252	14	0	14	176
breasttumor	286	9	8	1	23
cholesterol	303	13	7	6	152
cloud	108	6	2	4	94
concrete	103	9	0	9	83
cpu	209	7	1	6	104
dee	365	6	0	6	365
diabetes	43	2	0	2	20
echomonths	130	9	3	6	47
ele-1	495	2	0	2	453
ele-2	1056	4	0	4	1011
elusage	55	2	1	1	52
fishcatch	158	7	2	5	97
fruitfly	125	4	2	2	47
kidney	76	5	2	3	20
laser	993	4	0	4	191
lowbwt	189	9	7	2	133
machine	209	6	0	6	116
mbagrade	61	2	1	1	57
pbc	418	18	8	10	399
pharynx	195	11	10	1	177
pollution	60	15	0	15	60
pyrim	74	27	0	27	63
sensory	576	11	11	0	11
strike	625	6	1	5	358
veteran	137	7	4	3	101

For all experiments, unless otherwise stated in the description of the experiment, the result is obtained as an averaged value of the relative root squared error $RRSE$ over 10-fold cross-validation (see Section 2.6 for more information).

Table 5.2: Default values of the algorithms that are used for the the induction of regression rules.

parameter	default value	comment
$minCov$	3	minimal number of covered examples, fixed value
$maxNeighbours$	1	number of neighbours, bottom-up only, fixed value
$qualityMeasure$	-	tested parameter
$evaluationStrategy$	normal	tested parameter
$pruningStrategy$	heuristic	tested parameter
$filtrationStrategy$	none	tested parameter
$conflictResolutionStrategy$	mean	tested parameter

5.3. Quality measure for regression

Selecting the best quality measure to control the process of regression rule induction with the use of any algorithm is not a trivial problem. Tables 5.3, 5.4, 5.5 and 5.6 list the results of eight quality measures for 30 data sets with corresponding ranks in the superscripts (lower values are better) for Top-Down, Top-Down Fixed, Bottom-Up and Bottom-Up fixed algorithms respectively. In the last two rows of each table the averaged value of $RRSE$ over all data sets and averaged ranks are presented for each quality measure.

In each of the presented approaches the Friedman test indicates (see Table 5.7) significant differences between the compared quality measures and shows that the selection of the quality measure in the case of regression rule induction has a huge impact on the prediction error. Looking at the values of RRSE for each algorithm and quality measure (see Table 5.8) it can be noted that the highest (worst) prediction error was observed for the *BUF* algorithm. In contrast, the lowest (best) prediction error was observed for the *TDF* algorithm. Further analysis is based on the detection of the quality measure or groups of measures that achieve the smallest error value. All tests were performed at $p = 0.05$ where the critical value for $k = 8$ and $N = 30$ is 3.031 and the corresponding CD is once again 1.917.

In the case of the TD algorithm one can distinguish 4 groups. These groups are presented in Figure 5.1. It can be seen that the he first group consists of 4 measures, but only the LS measure is significantly better than all 4 measures on the left side. The first group is however

Table 5.3: Averaged RRSE for 8 selected quality measures on the Top-Down rule induction algorithm and 35 benchmark data sets. Values in superscripts correspond to the Friedman test rank value for each quality measure on one tested data set.

data set	C1	C2	RSS	Corr	LS	G	wLap	sBay
auto93	75.84^2	82.46^4	91.59^6	94.14^7	76.06^3	95.30^8	65.49^1	89.64^5
auto-mpg	61.12^3	66.90^4	68.81^5	70.02^6	52.12^1	74.34^7	53.63^2	80.95^8
auto-price	53.38^2	59.21^4	55.86^3	71.99^6	46.82^1	93.78^7	62.09^5	100.41^8
baseball	89.88^3	95.75^4	115.90^8	115.82^7	82.34^2	106.75^6	81.03^1	105.91^5
bodyfat	46.73^4	47.06^5	44.89^1	48.51^7	47.27^6	52.41^8	46.47^3	45.89^2
breasttumor	103.10^7	102.31^6	96.92^1	99.87^4	103.17^8	99.17^3	101.28^5	99.01^2
cholesterol	103.63^7	101.19^4	100.29^1	102.30^5	102.79^6	100.98^2	106.46^8	101.08^3
cloud	64.94^1	65.05^2	80.38^6	79.22^5	66.01^3	87.57^7	67.93^4	99.52^8
concrete	84.51^4	84.04^2	98.38^6	98.95^7	82.96^1	101.42^8	84.12^3	94.80^5
cpu	66.59^3	80.44^4	102.55^7	98.99^6	62.63^1	95.34^5	63.62^2	105.23^8
dee	58.89^2	60.15^5	64.24^6	59.52^3	55.73^1	66.63^7	59.95^4	96.32^8
diabetes	86.00^2	83.17^1	92.64^7	92.08^4	92.28^5	89.05^3	101.33^8	92.29^6
echomonths	79.55^5	77.99^4	73.91^1	74.58^2	81.33^6	83.19^8	81.58^7	77.55^3
ele-1	68.05^3	72.63^4	95.04^7	92.75^5	63.56^1	94.11^6	66.36^2	100.00^8
ele-2	45.12^3	49.23^5	47.97^4	56.70^6	41.10^1	90.44^7	42.35^2	106.80^8
elusage	57.28^3	69.32^5	69.50^6	68.20^4	54.73^2	85.72^7	53.13^1	86.54^8
fishcatch	52.69^2	60.18^5	57.37^4	66.31^6	53.10^3	105.00^8	50.52^1	91.00^7
fruitfly	106.12^8	105.93^7	105.10^5	104.13^2	105.07^4	102.75^1	105.87^6	104.29^3
kidney	107.47^6	105.40^5	109.44^7	112.38^8	105.17^4	101.82^3	98.75^1	100.11^2
laser	67.95^3	78.09^4	95.60^6	90.31^5	58.71^2	101.86^7	56.34^1	102.40^8
lowbwt	65.80^6	61.41^1	62.74^2	62.78^3	67.38^7	77.27^8	65.14^5	63.93^4
machine	65.62^2	70.28^4	103.26^7	96.27^5	61.33^1	97.70^6	65.96^3	106.12^8
mbagrade	91.77^3	91.27^2	96.51^8	93.41^6	91.79^4	93.18^5	94.60^7	89.70^1
pbc	90.38^1	93.17^5	93.64^6	92.57^4	91.49^3	98.47^8	91.25^2	98.32^7
pharynx	80.40^7	77.32^1	77.54^2	78.45^4	78.62^5	97.53^8	77.62^3	79.56^6
pollution	74.23^2	79.68^4	94.96^6	89.49^5	72.51^1	95.88^7	77.21^3	102.23^8
pyrim	74.11^2	75.49^3	91.67^6	91.11^5	73.36^1	91.85^7	83.29^4	99.23^8
sensory	91.89^2	90.23^1	96.50^7	92.82^6	91.93^3	98.36^8	92.43^5	92.02^4
strike	96.41^4	99.20^5	106.46^8	99.78^6	94.79^3	105.42^7	92.59^1	94.45^2
veteran	94.06^3	95.60^4	98.48^8	96.37^5	92.60^1	97.40^6	92.81^2	98.47^7
avg. RRSE	76.78	79.34	86.27	86.33	74.96	92.69	76.04	93.46
avg. rank	3.50	3.80	5.23	5.13	3.0	6.27	3.40	5.67

Table 5.4: Averaged RRSE for 8 selected quality measures on the Top-Down Fixed rule induction algorithm and 35 benchmark data sets. Values in superscripts correspond to the Friedman test rank value for each quality measure on one tested data set.

data set	C1	C2	RSS	Corr	LS	G	wLap	sBay
auto93	68.99^{3}	70.04^{4}	72.96^{5}	63.69^{1}	66.69^{2}	76.30^{7}	80.59^{8}	74.03^{6}
auto-mpg	54.81^{4}	54.25^{3}	55.11^{5}	56.69^{7}	53.81^{2}	55.46^{6}	49.55^{1}	59.77^{8}
auto-price	58.74^{2}	65.88^{8}	59.63^{4}	60.13^{6}	59.70^{5}	63.29^{7}	59.11^{3}	57.17^{1}
baseball	73.23^{3}	69.15^{1}	73.28^{4}	69.92^{2}	74.71^{5}	77.18^{6}	77.61^{7}	78.66^{8}
bodyfat	41.02^{5}	39.52^{4}	41.29^{6}	36.10^{3}	35.39^{2}	57.96^{8}	32.74^{1}	45.37^{7}
breasttumor	117.55^{5}	119.31^{6}	102.91^{1}	113.04^{4}	122.18^{7}	103.03^{2}	122.48^{8}	109.58^{3}
cholesterol	115.94^{4}	123.15^{7}	114.01^{2}	115.08^{3}	122.30^{6}	107.40^{8}	123.42^{1}	116.31^{4}
cloud	73.34^{7}	68.79^{5}	63.91^{2}	65.61^{3}	68.87^{6}	76.61^{8}	56.82^{1}	67.69^{4}
concrete	91.16^{8}	73.72^{2}	78.86^{5}	90.06^{7}	70.24^{1}	87.42^{6}	73.96^{4}	73.79^{3}
cpu	66.05^{5}	69.14^{6}	62.49^{2}	64.26^{3}	64.81^{4}	70.69^{7}	48.54^{1}	89.67^{8}
dee	49.51^{1}	51.47^{4}	50.41^{2}	53.33^{5}	50.63^{3}	61.79^{8}	54.61^{6}	57.42^{7}
diabetes	91.19^{1}	100.06^{5}	94.60^{2}	102.64^{7}	100.65^{6}	97.60^{3}	118.46^{8}	99.75^{4}
echomonths	81.82^{3}	84.64^{5}	77.70^{1}	87.21^{8}	85.91^{7}	83.16^{4}	85.79^{6}	81.39^{2}
ele-1	69.60^{4}	67.61^{3}	70.12^{5}	70.28^{6}	64.60^{2}	91.46^{8}	62.48^{1}	85.94^{7}
ele-2	45.80^{6}	39.45^{4}	38.96^{3}	53.64^{7}	34.99^{2}	61.71^{8}	23.01^{1}	41.81^{5}
elusage	60.46^{2}	62.40^{5}	55.25^{1}	65.30^{7}	70.71^{8}	64.26^{6}	62.22^{4}	60.54^{3}
fishcatch	62.23^{6}	54.99^{4}	45.04^{1}	53.77^{3}	57.72^{5}	76.77^{8}	46.19^{2}	66.09^{7}
fruitfly	146.12^{7}	143.76^{6}	126.25^{3}	118.30^{2}	142.90^{5}	116.21^{1}	151.96^{8}	134.36^{4}
kidney	108.73^{6}	111.77^{7}	97.29^{1}	103.54^{3}	108.51^{5}	103.31^{2}	119.03^{8}	105.71^{4}
laser	59.34^{4}	61.93^{7}	53.79^{3}	59.95^{5}	46.38^{2}	81.67^{8}	41.21^{1}	61.75^{6}
lowbwt	70.03^{3}	69.37^{2}	70.25^{4}	66.27^{1}	73.96^{5}	83.98^{8}	76.49^{7}	75.25^{6}
machine	67.63^{5}	70.85^{6}	65.11^{4}	61.34^{3}	60.31^{2}	73.10^{8}	56.06^{1}	71.21^{7}
mbagrade	109.55^{7}	109.14^{6}	98.12^{2}	95.36^{1}	107.20^{5}	113.44^{8}	100.24^{4}	98.78^{3}
pbc	91.06^{4}	89.16^{2}	91.95^{5}	86.27^{1}	95.47^{6}	89.70^{3}	102.95^{7}	104.04^{8}
pharynx	83.03^{1}	83.91^{2}	88.09^{5}	84.06^{4}	98.17^{7}	83.98^{3}	100.25^{8}	90.70^{6}
pollution	80.72^{2}	77.34^{1}	95.01^{6}	91.23^{4}	95.34^{7}	104.21^{8}	93.42^{5}	86.96^{3}
pyrim	72.73^{3}	63.92^{1}	77.00^{4}	82.46^{6}	71.13^{2}	86.38^{8}	81.39^{5}	82.62^{7}
sensory	103.31^{1}	108.31^{5}	110.90^{6}	113.73^{7}	107.04^{3}	104.12^{2}	107.54^{4}	114.48^{8}
strike	88.58^{3}	89.99^{4}	105.63^{8}	95.04^{7}	85.71^{1}	92.57^{5}	86.26^{2}	92.79^{6}
veteran	106.82^{4}	106.42^{3}	106.13^{2}	116.56^{8}	99.35^{1}	108.38^{5}	115.78^{7}	108.86^{6}
avg. RRSE	80.30	79.98	78.07	79.83	79.85	85.10	80.34	83.08
avg. rank	3.97	4.27	3.47	4.47	4.13	5.73	4.57	5.40

Table 5.5: Averaged RRSE for 8 selected quality measures on the Bottom-Up rule induction algorithm and 35 benchmark data sets. Values in superscripts correspond to the Friedman test rank value for each quality measure on one tested data set.

data set	C1	C2	RSS	Corr	LS	G	wLap	sBay
auto93	74.81^4	75.19^5	77.43^6	74.79^3	70.06^1	84.63^8	72.63^2	81.51^7
auto-mpg	56.62^2	58.80^5	64.86^6	66.36^7	55.84^1	57.44^3	57.55^3	75.73^8
auto-price	57.27^2	61.43^5	57.74^3	62.34^6	61.16^4	66.84^8	55.06^1	66.44^7
baseball	80.30^2	86.23^4	100.10^6	100.12^7	76.15^1	102.71^8	82.68^3	99.13^5
bodyfat	37.19^1	37.71^2	45.57^6	43.72^4	42.77^3	46.65^7	44.67^5	54.36^8
breasttumor	99.54^8	99.51^7	98.75^3	97.94^1	98.73^2	98.84^4	99.25^6	99.17^5
cholesterol	108.00^5	111.00^8	101.62^2	103.91^3	108.50^6	99.92^1	109.66^7	107.71^4
cloud	69.67^1	73.51^4	80.55^5	83.67^6	70.55^2	99.85^7	72.79^3	100.76^8
concrete	74.60^3	72.20^2	76.87^5	79.44^6	76.06^4	85.42^7	69.95^1	102.02^8
cpu	57.66^4	56.53^3	61.43^5	62.63^6	54.01^2	101.51^8	52.01^1	88.78^7
dee	55.42^2	55.39^1	58.87^6	61.50^7	57.89^4	56.83^3	57.95^5	69.48^8
diabetes	97.91^6	95.56^3	100.21^8	98.56^7	96.49^5	95.89^4	90.74^1	94.17^2
echomonths	102.31^7	103.10^8	88.82^2	94.73^3	101.76^6	87.75^1	98.11^4	99.08^5
ele-1	64.96^3	71.74^4	83.50^5	89.06^6	60.45^1	97.90^7	60.61^2	100.55^8
ele-2	37.31^3	44.45^4	47.49^5	51.97^6	22.39^2	98.88^7	14.48^1	104.44^8
elusage	56.98^2	60.40^6	60.89^7	57.68^4	55.89^1	94.49^8	57.64^3	58.01^5
fishcatch	85.49^3	87.04^5	77.90^1	84.16^2	88.70^6	85.50^4	89.29^7	91.31^8
fruitfly	107.86^8	105.75^6	103.36^4	102.22^3	107.41^7	101.63^2	103.57^5	100.63^1
kidney	97.79^4	96.34^2	98.59^6	97.41^3	99.13^7	98.34^5	94.03^1	101.00^8
laser	41.46^1	52.03^4	69.62^6	68.31^5	42.03^2	103.70^8	42.27^3	103.07^7
lowbwt	71.41^4	66.47^1	66.65^3	66.62^2	73.29^5	77.56^7	75.11^6	86.40^8
machine	55.96^4	53.21^2	60.78^6	57.26^5	49.57^1	95.68^8	55.67^3	88.01^7
mbagrade	100.05^3	101.59^7	100.79^6	97.98^2	106.81^8	97.63^1	100.13^4	100.56^5
pbc	93.53^4	94.09^5	90.12^1	91.77^3	96.62^8	91.01^2	95.26^7	94.35^6
pharynx	88.18^6	82.85^3	91.09^7	81.34^2	81.09^1	99.20^8	88.14^5	88.01^4
pollution	88.84^4	90.42^5	80.95^2	79.81^1	88.77^3	97.07^8	95.94^7	92.79^6
pyrim	86.40^4	90.50^5	84.26^2	97.11^6	79.06^1	97.90^7	84.75^3	101.40^8
sensory	96.41^5	98.17^7	94.04^2	93.45^1	96.06^3	99.86^8	96.23^4	97.48^6
strike	94.57^2	97.61^4	107.65^7	102.85^5	95.34^3	107.99^8	93.00^1	106.10^6
veteran	97.78^4	98.23^5	100.89^8	99.02^7	97.14^1	97.76^3	97.17^2	98.90^6
avg. RRSE	77.88	79.23	81.05	81.59	76.99	90.88	76.88	91.71
avg. rank	3.70	4.40	4.70	4.30	3.37	5.67	3.57	6.30

Table 5.6: Averaged RRSE for 8 selected quality measures on the Bottom-Up Fixed rule induction algorithm and 35 benchmark data sets. Values in superscripts correspond to the Friedman test rank value for each quality measure on one tested data set.

data set	C1	C2	RSS	Corr	LS	G	wLap	sBay
auto93	75.79^{3}	75.97^{4}	73.35^{1}	77.29^{8}	76.36^{5}	74.84^{2}	76.50^{7}	76.39^{6}
auto-mpg	61.46^{4}	59.52^{1}	60.94^{3}	61.63^{5}	126.60^{7}	60.60^{2}	138.64^{8}	98.58^{6}
auto-price	65.45^{4}	59.69^{3}	58.08^{2}	54.37^{1}	157.31^{7}	69.55^{5}	171.42^{8}	91.43^{6}
baseball	88.48^{4}	86.68^{3}	89.95^{5}	84.72^{1}	142.75^{7}	85.49^{2}	149.09^{8}	97.06^{6}
bodyfat	45.28^{2}	44.17^{1}	54.05^{5}	51.92^{3}	58.81^{8}	57.89^{7}	52.62^{4}	54.16^{6}
breasttumor	134.63^{5}	112.66^{3}	104.03^{1}	106.94^{2}	139.67^{6}	114.34^{4}	145.23^{7}	162.63^{8}
cholesterol	122.95^{2}	124.16^{3}	134.94^{4}	136.96^{6}	137.59^{7}	106.80^{1}	135.02^{5}	153.60^{8}
cloud	67.94^{3}	65.81^{2}	68.07^{4}	69.87^{5}	74.50^{8}	64.25^{1}	72.77^{7}	71.59^{6}
concrete	85.29^{4}	79.04^{2}	97.48^{7}	95.52^{6}	80.12^{3}	88.36^{5}	68.52^{1}	98.19^{8}
cpu	69.47^{3}	67.05^{2}	71.96^{5}	66.65^{1}	170.09^{7}	84.98^{6}	327.32^{8}	71.88^{4}
dee	51.46^{1}	51.63^{2}	55.09^{4}	57.30^{6}	55.21^{5}	54.71^{3}	68.13^{7}	101.84^{8}
diabetes	122.36^{8}	114.59^{7}	113.01^{6}	103.58^{2}	107.43^{4}	105.55^{3}	99.09^{1}	107.64^{5}
echomonths	104.07^{4}	110.85^{5}	92.48^{2}	85.47^{1}	151.03^{7}	102.03^{3}	152.52^{8}	136.60^{6}
ele-1	67.48^{3}	73.41^{4}	66.86^{2}	75.24^{5}	66.72^{1}	79.23^{7}	75.39^{6}	85.82^{8}
ele-2	31.67^{2}	36.36^{4}	28.28^{1}	34.19^{3}	49.02^{7}	36.38^{5}	90.20^{8}	43.13^{6}
elusage	78.46^{7}	54.63^{2}	50.85^{1}	58.73^{3}	75.82^{6}	60.46^{4}	97.21^{8}	62.26^{5}
fishcatch	85.55^{4}	86.46^{5}	77.13^{1}	79.48^{2}	87.01^{6}	87.73^{7}	87.84^{8}	85.19^{3}
fruitfly	133.57^{6}	124.76^{4}	123.73^{3}	124.91^{5}	226.27^{8}	109.88^{2}	176.48^{7}	106.89^{1}
kidney	90.35^{2}	89.61^{1}	$92.48^{3.5}$	$92.48^{3.5}$	161.29^{7}	99.27^{5}	190.05^{8}	99.40^{6}
laser	51.23^{2}	47.54^{1}	58.96^{3}	64.96^{4}	153.25^{7}	87.39^{5}	185.14^{8}	87.74^{6}
lowbwt	76.55^{4}	70.75^{1}	76.15^{3}	72.87^{2}	83.49^{6}	80.50^{5}	83.97^{7}	113.75^{8}
machine	53.78^{1}	70.40^{6}	67.96^{4}	67.06^{3}	69.41^{5}	78.09^{7}	181.64^{8}	62.75^{2}
mbagrade	105.16^{2}	96.79^{1}	115.83^{4}	120.23^{6}	138.56^{8}	105.90^{3}	133.43^{7}	118.44^{5}
pbc	95.81^{4}	98.32^{5}	89.06^{1}	89.83^{2}	100.60^{6}	91.41^{3}	102.00^{8}	100.94^{7}
pharynx	106.40^{5}	86.68^{1}	93.28^{2}	96.60^{3}	180.17^{7}	97.98^{4}	185.81^{8}	157.33^{6}
pollution	77.15^{1}	81.47^{4}	78.18^{2}	79.82^{3}	93.18^{8}	86.77^{7}	86.34^{6}	85.89^{5}
pyrim	86.83^{3}	87.05^{4}	80.12^{1}	82.66^{2}	172.33^{8}	96.14^{5}	168.93^{7}	103.92^{6}
sensory	129.27^{4}	115.13^{3}	135.19^{5}	112.40^{2}	153.64^{8}	104.51^{1}	143.00^{6}	146.32^{7}
strike	90.38^{2}	91.14^{3}	108.55^{8}	101.28^{6}	89.87^{1}	95.94^{5}	93.60^{4}	107.66^{7}
veteran	101.77^{1}	107.97^{6}	102.82^{3}	102.02^{2}	104.24^{4}	106.67^{5}	113.35^{7}	114.27^{8}
avg. RRSE	85.20	82.34	83.96	83.57	116.08	85.79	128.38	100.11
avg. rank	3.33	3.10	3.22	3.45	6.13	4.13	6.67	5.97

Table 5.7: Comparative table of the results of statistical tests of 8 quality measures over 30 data sets for each of the algorithms. The critical value for the Friedman test is 2.2662. The plus sign next to the value of the Friedman test indicates a significant difference between quality measures.

algorithm	Friedman test	p-value
TD	9.51^{+}	$6.2340 \cdot 10^{-8}$
TDF	2.95^{+}	$7.0525 \cdot 10^{-3}$
BU	6.29^{+}	$3.8823 \cdot 10^{-6}$
BUF	17.33^{+}	$2.7089 \cdot 10^{-14}$

Table 5.8: Average value of RRSE for all algorithms and selected quality measures on 30 benchmark data sets.

	TD	TDF	BU	BUF
C1	76.78	80.30	77.88	85.20
C2	79.34	79.98	79.23	82.34
Correlation	86.33	79.83	81.59	83.57
RSS	86.27	78.07	81.05	83.96
G (g=2)	92.69	85.10	90.88	85.79
wLap	76.04	80.34	76.88	128.38
LS	74.96	79.85	76.99	116.08
sBay	93.46	83.08	91.71	100.11
average	83.23	80.82	82.03	95.68

more uniform than in the previous experiment because all quality measures which belong to this group (LS, $wLap$, $C1$ and $C2$) induce a greater number of rules (see Table 5.9) at the cost of lower average coverage (Table 5.10).

In the TDF algorithm the Nemenyi test indicates only 2 groups of quality measures (Figure 5.2), but only the RSS measure allows to obtain the significantly lower prediction error than $g - measure$ or $sBayesian\ confirmation\ measure$. Among other measures it is not possible to indicate the statistically significant difference. Thus, the choice of a quality measure with the given default settings for the remaining parameters has no statistical impact on the prediction error.

The Nemenyi test for the BU algorithm allowed to identify 3 groups of quality measures (Figure 5.3) with the lack of a statistically significant difference between measures. The $sBayesian$ and g measures again are worse than the three quality measures that obtained the lowest prediction error (LS, $wLap$ and $C1$). However these 3 measures are indistinguishable with $Correlation$, $C2$ and RSS.

A comparative analysis between TD and BU algorithm indicates that 3 best measures for both algorithms are in exactly the same order and the order of the remaining measures is also very similar (only the exchange of positions between neighbouring measures can be seen). This observation may seem somewhat surprising because of the very different nature of the algorithms for rule induction. However, it should be noted that the distance between the two middle measures in the TD algorithm is 1.33 while in the BU algorithm this gap does not really exist (0.1), as evidenced by a smaller impact on the choice of the measurement prediction error in the latter case.

The most interesting experiments are for the last algorithm (BUF). At $p = 0.05$ there are 3 indistinguishable, in terms of the value of the resulting errors, groups of quality measures. The first group of measures (with $C2$, RSS, $C1$, $Correlation$ and G measures) outperforms the $wLap$ and LS measures whereas the $sBayesian$ measure does not differ only from the G measure. The second group contains only two measures: $sBayesian$ and G. The last group consists of the three worst measures: $wLap$, LS and $sBayesian$. However, this CD diagram is different when the level of confidence is decreased to $p = 0.10$ with the corresponding value of $CD = 1.7582$. It may be noted that the diagram has only two completely separated groups of quality measures. This implies that quality measures from this group allow to obtain a statistically significant lower error value of the prediction.

Table 5.9 presents the average number of rules and average number of elementary conditions for each rule obtained for the tested algorithms in relation to various quality measures. The table shows that the number of rules depends on both the selected rule induction algorithm and the chosen quality measure. Referring to the result of the prediction error of each measurement,

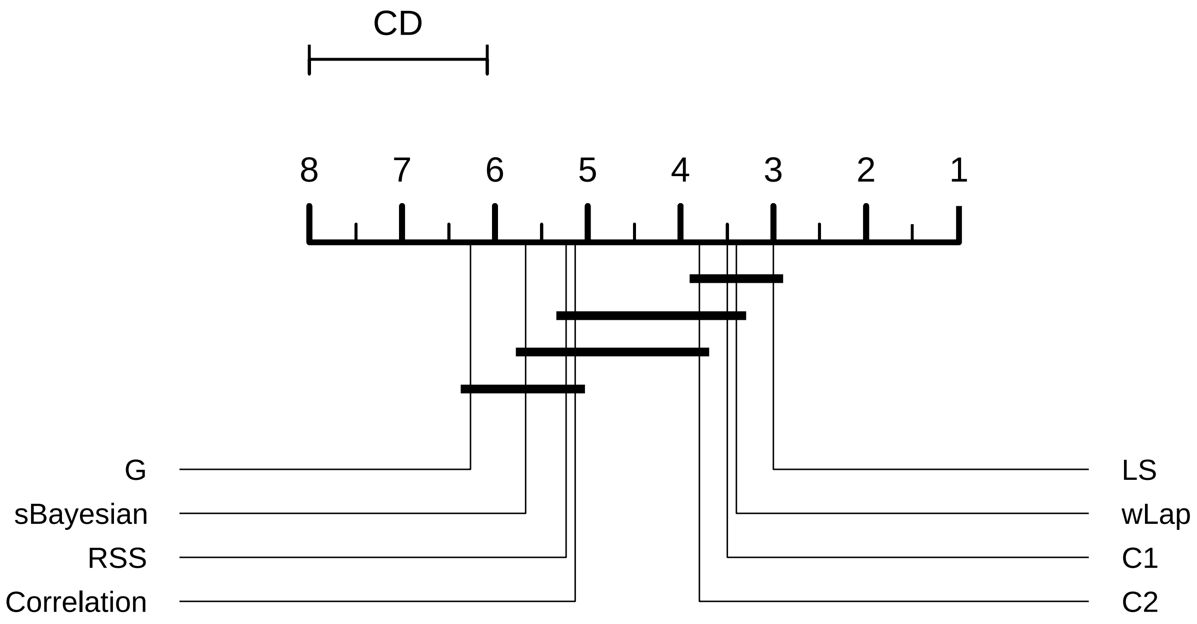


Figure 5.1: Comparison of quality measures against each other with the Nemenyi test at $\alpha = 0.05$ for the TD algorithm only.

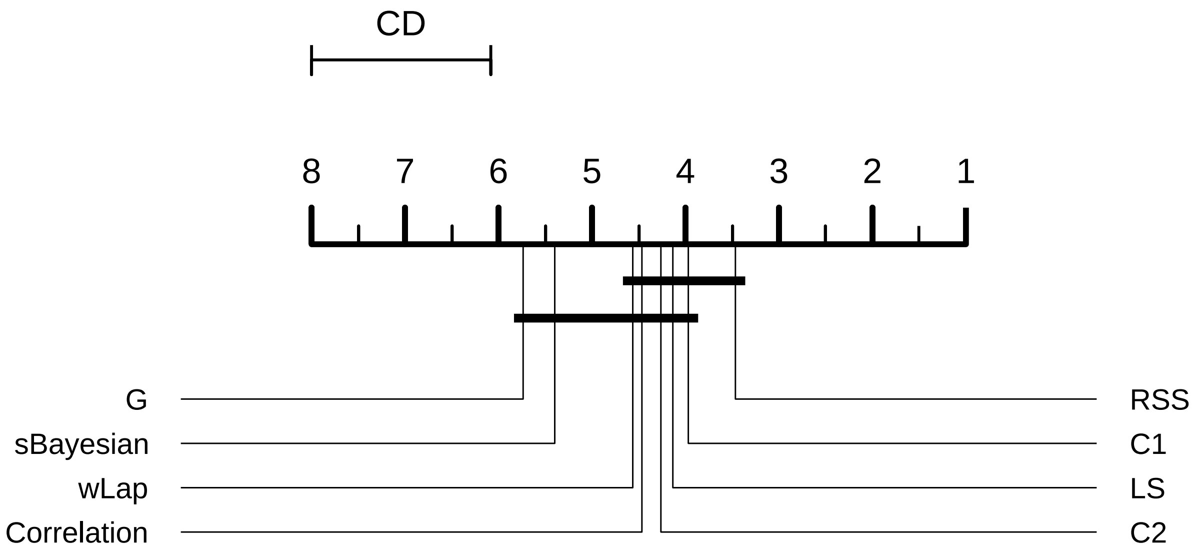


Figure 5.2: Comparison of quality measures against each other with the Nemenyi test at $\alpha = 0.05$ for the TDF algorithm only.

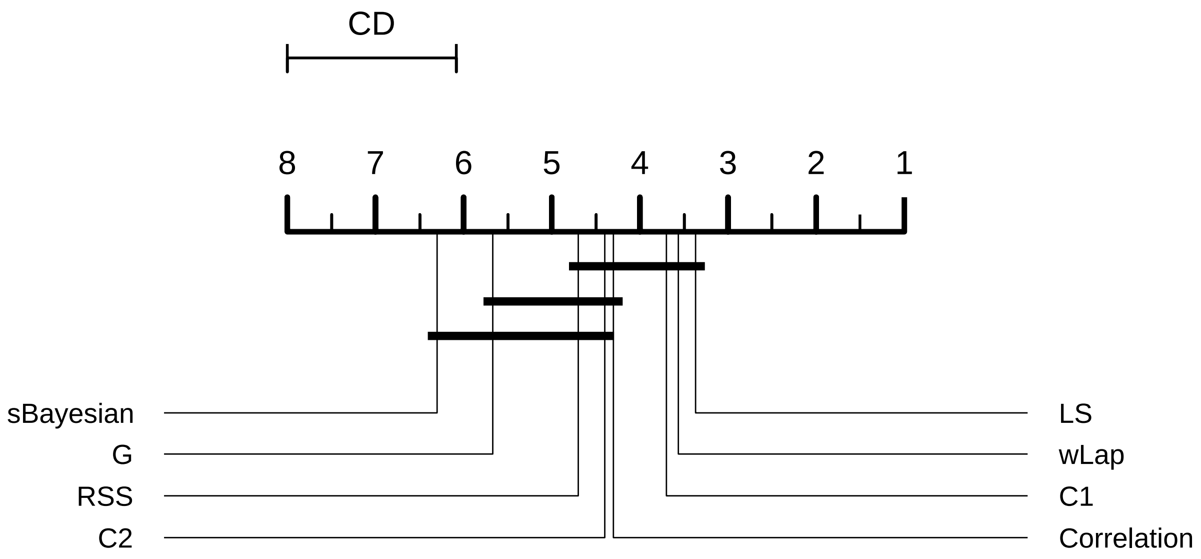


Figure 5.3: Comparison of quality measures against each other with the Nemenyi test at $\alpha = 0.05$ for the BU algorithm only.

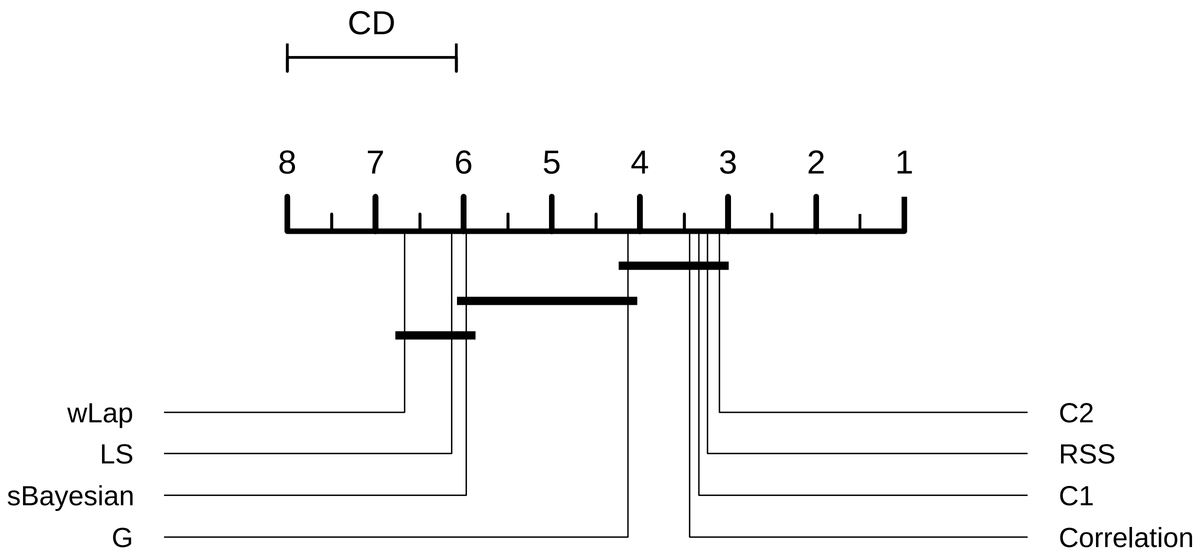


Figure 5.4: Comparison of quality measures against each other with the Nemenyi test at $\alpha = 0.05$ for the BUF algorithm only.

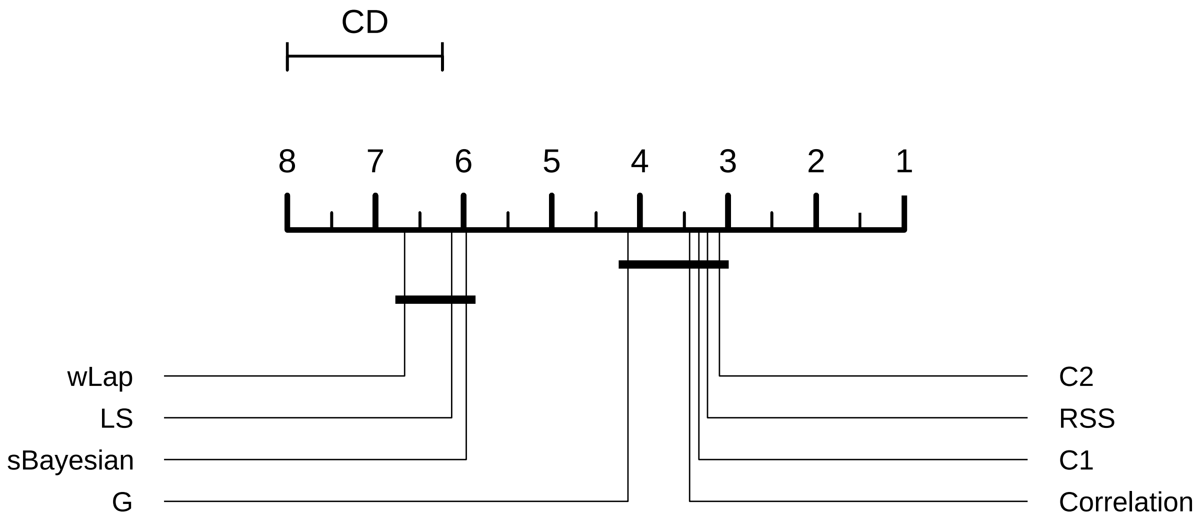


Figure 5.5: Comparison of quality measures against each other with the Nemenyi test at $\alpha = 0.1$ for the BUF algorithm only.

Table 5.9: Average number of rules and elementary conditions (in parentheses) for all algorithms and selected quality measures on 30 benchmark data sets.

	TD	TDF	BU	BUF
C1	97.18 (2.48)	40.87 (3.53)	28.94 (5.03)	23.25 (5.09)
C2	74.71 (2.55)	31.08 (3.73)	19.85 (5.04)	18.03 (5.12)
Correlation	46.34 (2.50)	15.97 (4.16)	11.05 (4.82)	7.43 (4.94)
RSS	50.40 (2.16)	13.22 (3.49)	10.16 (4.38)	6.99 (4.47)
G (g=2)	42.82 (2.32)	23.53 (3.47)	10.43 (5.19)	10.16 (5.04)
wLap	176.06 (2.23)	82.06 (3.33)	45.22 (4.82)	37.33 (4.87)
LS	144.58 (2.32)	60.42 (3.38)	48.01 (4.78)	35.31 (4.89)
sBay	36.29 (2.81)	29.90 (4.05)	10.97 (5.07)	9.41 (4.98)

one can say that the most accurate (the smallest error) prediction is performed for the quality measures which tend to return bigger number of rules ($C1$, $C2$). However, taking into account the prediction error and the number of rules at the same time, the $Correlation$ and RSS measures seem to be the best choice. This was also proved by our another experiment, which has been published in 2012 [109]. The additional information is presented by the average number of conditions in one rule. It can be noted that the average number of conditions depends more on different rule the induction algorithms than on quality measures. The most general rules were induced for the *TD* algorithm, while more specific rules were generated for both the *BU* and *BUF* algorithms. Referring to the quality measures it can be seen that the $C1$ and $C2$ measures not only return a bigger number of rules but those rules are also more complex with respect to other quality measures. In turn, the $Correlation$ measure allows to obtain rules with similar number of conditions like in the case of $C1$ and $C2$ measures, but the number of rules is smaller.

Table 5.10: Average coverage of rules for all algorithms and selected quality measures on 30 benchmark data sets.

	TD	TDF	BU	BUF
C1	0.185	0.147	0.147	0.169
C2	0.248	0.190	0.217	0.217
Correlation	0.354	0.278	0.364	0.364
RSS	0.392	0.332	0.400	0.387
G (g=2)	0.363	0.266	0.513	0.371
wLap	0.115	0.085	0.117	0.154
LS	0.124	0.103	0.108	0.150
sBay	0.364	0.175	0.501	0.399
average	0.268	0.197	0.296	0.276

Table 5.10 shows the average coverage of rules that is defined as a ratio of the number of training examples covered by the rule to the number of all training examples. It can be observed that the coverage of rules differs depending on the rule induction algorithm and the quality measures used to control the process of the induction. The highest coverage was provided by $Correlation$, RSS, $g - measure$ and $sBayesian$. In turn, the lowest coverage was observed for $WeightedLaplace$, $LogicalSufficiency$ and $C1$. Referring to the algorithm of induction, the highest coverage was observed for the *BU* algorithm, however the values obtained for *TD* and *BUF* are very similar. The lowest coverage was observed for the *TDF* algorithm. It can be also noted that for both fixed versions of the algorithm the observed values are lower than in their standard counterparts.

The results lead us to conclusions that the two measures may provide interesting results: $C2$ and $Correlation$. First of all, the choice of $C2$ was dictated by the overall results of the comparison measurement experiment (one of the lowest RRSE and a smaller number of rules or better average coverage than for the $C1$ measure). In contrast, the $Correlation$ measure will be used as a compromise between the prediction error, the number of rules and one of the highest coverages. Both quality measures allow to return more specific rules, which can be observed in the number of elementary conditions. Accordingly, these two measures were selected for further experiments.

5.4. Confidence intervals for examples covered by a rule

As we mentioned before, the quality of the rule, which is induced on the basis of the data sample, is subjected to the distribution of this sample. Therefore, in the perfect case the algorithm should consider entire population to prevent this problem. Otherwise, the statistical significance of the results should be assessed. In some cases, e.g. in knowledge discovery experiments, the best rules with respect to precision and coverage should be chosen as the most interesting. Therefore, the minimal precision and minimal coverage threshold can be set to choose the best rules. When the pessimistic approach is utilized fewer rules will meet the threshold and the analyzed rule set will be smaller and, therefore, more easily interpretable. While in the case of a resulting rule set being too small, the optimistic approach can be utilized, which will result in a larger number of rules meeting the threshold.

In this section we would like to present an approach to assess rules, which can be used to build a decision support system to assist the users of monitoring systems. The evaluation of statistical significance of rules may also be carried out by an expert in various types of supervise or knowledge discovery applications.

The data used in this experiment are derived from the *TD* algorithm on the *bodyfat* data set. The rules were induced with the use of *C2* heuristic. The statistical analysis is performed based on pessimistic, optimistic or standard quality of the rules as it was discussed in Section 3.4. In this experiment we decided to present ways to draw conclusions based on the values of precision, coverage and accuracy quality. However, it should be noted that any quality measure can be calculated this way, e.g., the C2 measure.

	Rule R1
IF	Density $\geq$ 1.064
THEN	class = 10.15 (3.843)

Table 5.11: Exemplary rule generated by means of the TDF method.

			Rule quality	
		pessimistic	standard	optimistic
R1	Precision	0.493	0.654	0.802
	Coverage	0.725	1.000	1.000

Table 5.12: Quality of the rule from Table 5.11.

Table 5.11 presents the example of a rule generated by means of the *TDF* algorithm and Table 5.12 shows the statistics obtained for this rule. The standard coverage of the rule is perfect, but the precision tells that only 65.4% of examples covered by this rule are within the range of conclusion $\pm$ the standard deviation. In the optimistic case, the precision of this rule increases, but in the pessimistic case, the rule covers more negative than positive examples (pessimistic precision is < 0.5). Therefore one may conclude that this rule is not very reliable.

Tables 5.13 and 5.14 present the second exemplary case of generated rules involving the *TDF* algorithm and their corresponding statistics. It should be noted that all rules are very

	Rule R2		Rule R3		Rule R4
IF	Density $<$ 1.049	IF	Density $<$ 1.057	IF	Density $<$ 1.042
	Wrist $<$ 19.05		Wrist $<$ 18.9		Wrist $<$ 19.95
THEN	26.35 (4.312)	THEN	24.4 (5.109)	THEN	29 (3.952)

Table 5.13: Exemplary rules generated by means of the TDF method - case 2.

			Rule quality	
		pessimistic	standard	optimistic
	Precision	0.679	0.833	0.955
R2	Coverage	0.5625	0.781	1.000
	Precision	0.631	0.767	0.885
R3	Coverage	0.602	0.784	0.966
	Precision	0.783	0.909	1.000
R4	Coverage	0.692	0.962	1.000

Table 5.14: Quality of the rules from Table 5.13.

similar in the bodies (the same attributes were selected and the same operator, while the value of attributes differs by thousandths or decimal parts), but different in the heads. This result may lead to the first conclusion, which is that the small difference in the values of the *Density* and *Wrist* attributes seems to be crucial in the selection of examples covered by the rule and conclusions assigned to them. Table 5.14 allows to draw the further conclusions. It can be noticed that all rules are characterized by different precisions. The highest standard precision was observed for the rule *R4* and the lowest for the rule *R3*, but looking at the ranges derived by pessimistic and optimistic values we can also notice that the range for the rule *R2* is much broader than for the rules R3 and *R4*, thus the rule *R3* seems to be more stable than *R2* but less stable than *R4*. The standard coverage shows that the rules *R2* and *R3* are very similar but, once again, the ranges derived by pessimistic and optimistic values show that the rule *R3* covers more examples from the range of conclusion $\pm$ standard deviation. In general, the highest standard, pessimistic and optimistic values of coverage were obtained for the rule *R4*, which outperforms other rules in precision and coverage at the same time. The last measure (accuracy) holds a small amount of information but allows to say more about the number of covered examples. The highest standard and optimistic accuracy was observed for the rule *R3* before *R4* and *R2*. However, this rule is also characterized by a broader range of optimistic and pessimistic values. The narrowest range is showed in the rows assigned to the rule *R4* and thus it proves that this rule has the best quality (values of precision, coverage and accuracy). The rule *R4* seems to be the most accurate, stable and reliable.

	Rule R5		Rule R6
IF	Density < 1.056	IF	Density < 1.058
	Knee < 40.1		Biceps < 32.05
			Knee < 38.95
THEN	23.2 (5.098)	THEN	22.55 (4.640)

Table 5.15: Exemplary rules generated by means of the TDF method - case 3.

		Rule quality		
		pessimistic	standard	optimistic
	Precision	0.662	0.806	0.924
R5	Coverage	0.478	0.644	0.811
	Accuracy	21	44	67
	Precision	0.704	0.882	1.000
R6	Coverage	0.221	0.349	0.477
	Accuracy	11	26	41

Table 5.16: Quality of the rules from Table 5.15.

Table 5.15 shows a different case of the rule analysis. The rule *R5* consists of two attributes, while the rule *R6* consists of three attributes, but two of them are identical to the attributes from the rule *R5*. The conclusions in the heads of the rules are different, therefore it seems that the rule *R6* could be more specific, because of this additional elementary condition. To compare the rules, the confidence interval approach can be used. Table 5.16 presents the result of this comparison. The standard value of precision for the rule *R6* is a bit better than for the rule *R5*. Looking at the ranges derived by pessimistic and optimistic values we can notice that the range of rule *R6* precision is broader than of the rule *R5* precision, while all values are higher in the rule *R6* than their counterparts in the rule *R5*. The rule *R6* is then more precise. However, looking at the values of coverage one can notice a significant difference between the rules. The values of coverage for the rule *R5* are significantly higher than for the rule *R6* and it is worth noticing that the pessimistic value of coverage for the rule *R5* is higher than the optimistic value of coverage for the rule *R6*. It means that in relation to the coverage measure the rule *R5* outperforms the rule *R6*. This reflection and the value of accuracy confirm that the rule *R5* is more general. This investigation leads us to the conclusions that the rule *R5* has better quality (in the meaning of a compromise between values of precision, coverage and accuracy) and seems to be a better choice.

The presented examples of the pessimistic and optimistic rule quality analysis show that it can bring additional knowledge about the generated rules. It could be useful both to extract information about a single rule or to compare two rules that are similar or in the parent-child schema. This approach can be then used to build a support system to assist in any decision-making process.

5.5. Pre-pruning methods evaluation

The choice of the right pre-pruning method to prevent the overfitting problem during the process of regression rules induction is as important as the induction algorithm. In our experiment we would like to compare two approaches: the simplest *Hill climbing* method and the more sophisticated *Tabu hill climbing* method.

However, before the results are presented it is worth recalling (see Section 4.1) that pre-pruning algorithms are strongly related to the separate-and-conquer implementation, thus, in most cases the heuristic for the rule evaluation is defined in the same way as in the growing phase. The state of the art approach for the classification problem, however, proposes a phase separation of rule refinements and rule selection [115], undermining the uniform heuristic approach. This thesis was supported by an erroneous approach to the topic, where the optimization process focuses on selecting the currently best rule instead of selecting the best rule for further refinement.

These assumptions motivate us to check two theses. Firstly, various pre-pruning methods may produce different rules and therefore can lead to a different rule-based model. Secondly, apart from the heuristic which is used to control the process of growing the rule, the rule can be pruned using the same or another quality measure. In this second case, the characteristics of both measures could be completely different, e.g. one heuristic tends to return the more accurate (prediction abilities) while the other focuses on the more transparent model (descriptive abilities). Anticipating the next chapter, here one such pair of quality measures is used. However, the motivation to choose this pair will be presented in the next chapter. In addition, it is worth considering this section (Section 5.5) and the next one (see Section 5.6) as a unit and the continuation of the experiments presented here.

The efficiency of two pre-pruning methods was verified by three pairs of quality measures, which were used for the induction and pre-pruning respectively: ($C2$, $C2$), ($Correlation$, $Correlation$) and ($C2$, $Correlation$) as the choice of a more accurate pair of two previously tested mixtures.

The Friedman test indicates (see Table 5.17) significant differences between pre-pruning methods and quality measures only in both variants of the Top-Down algorithm. In the case of Bottom-Up and Bottom-Up Fixed algorithms there are no statistical significant differences, thus the further analysis would concern only the significant results. These results are presented in Figure 5.6 and Figure 5.7.

Table 5.17: Comparative table of the results of statistical tests against the difference between 2 pre-pruning methods and 3 pairs of quality measures over 30 data sets for each algorithm. The critical value for the Friedman test is 2.4205. The plus sign near the value of the Friedman test indicates the significant difference between quality measures.

algorithm	Friedman test	p-value
TD	9.16^{+}	$9.4523 \cdot 10^{-7}$
TDF	2.90^{+}	0.0181
BU	1.90	0.1007
BUF	1.86	0.1081

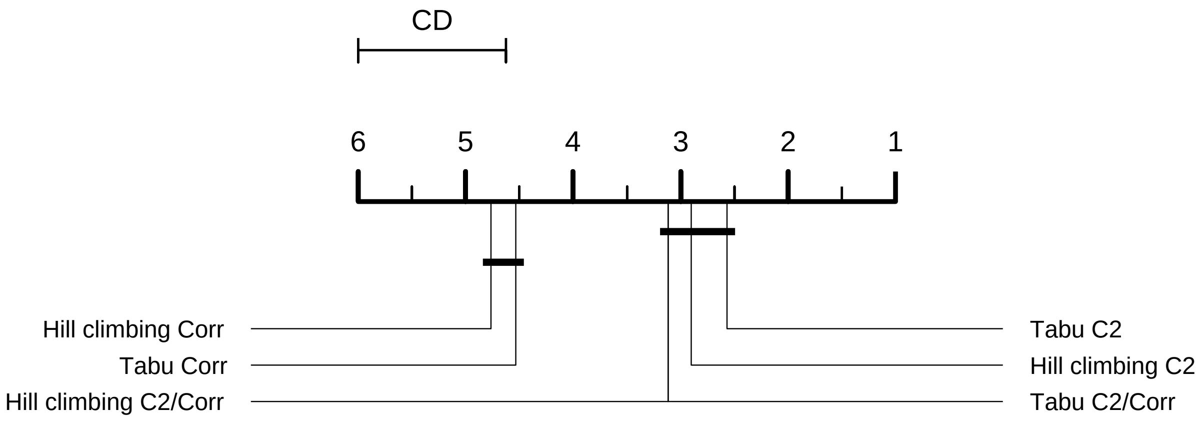


Figure 5.6: Comparison of 2 pre-pruning methods for 3 pairs of quality measures, which were used to induce (the first measure) and to prune (the second measure) rule using the Top-Down algorithm. Groups of combination that are not significantly different (at $p = 0.05$) are connected.

In the *TD* algorithm the post-hoc Nemenyi test at 0.05 indicates 2 separated groups. The first group consists of results obtained for the heuristic and tabu algorithm using the $Correlation$ measure while the second group is built of results for both pre-pruning methods using combinations of the $C2/Correlation$ and $C2/C2$ measures. However, there are no significant differences between presented pre-pruning methods. In the case of the *TDF* algorithm the results are similar. The other experiment can check the hypothesis whether the *Tabu hill climbing* algorithms on average perform statistically better than the simplest *Hill climbing* algorithm or not in the *TD* or *TDF* algorithm. The easiest way to verify this hypothesis is to group the results obtained in the previous experiments and then to check the hypothesis using the Wilcoxon test. However, for the presented result the Wilcoxon signed ranks test at $\alpha = 0.05$ shows that the null-hypothesis cannot be rejected (with p-value=0.0679 for the *TD* algorithm and with p-value 0.7150 for the *TDF* algorithm), which means that pre-pruning methods cannot be statistically distinguished in any of 4 rule induction algorithms.

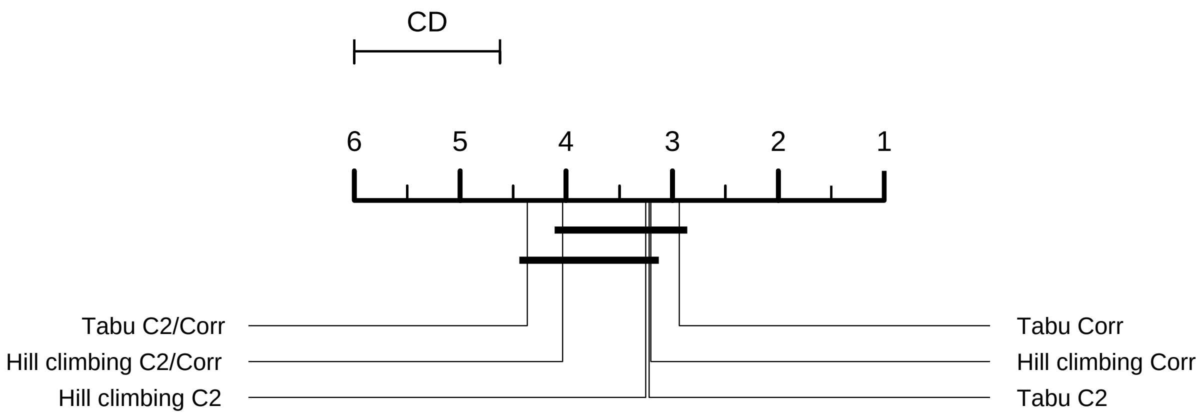


Figure 5.7: Comparison of 2 pre-pruning methods for 3 pairs of quality measures, which were used to induce (the first measure) and to prune (the second measure) rule using the Top-Down Fixed algorithm. Groups of combination that are not significantly different (at $p = 0.05$) are connected.

Pruning methods may also be assessed on the basis of the average number of rules in the final model. However, it is important to mention that the number of rules in the case of the *Tabu hill climbing* method may depend on the number of steps (tabu elements) that the algorithm needs to remember. Let us consider an example. Assume that the *Tabu hill climbing* algorithm remembers only one best elementary condition in each iteration. In this case it is possible to remove all but one elementary conditions, whereas such a rule would be probably the most general and would cover more examples than if the rule would be specific. This implies a smaller number of rules needed to cover the whole set. The examination of the optimal number of steps, however, goes beyond the scope of this work.

In our implementation we do not define this parameter, which means that the algorithm can crop the rule up to a half of all elementary conditions (under the condition that the quality of the rule improves with each removal). Thus, this approach should provide an optimal construction of the rules and an appropriate coverage.

Figures 5.8 and 5.9 present an average number of rules for the *Tabu hill climbing* and *Hill climbing* methods of pruning for the *TD* and *TDF* algorithms respectively. All values are an average over three selected pairs of heuristic: ($C2$, $C2$), ($Correlation$, $Correlation$) and ($C2$, $Correlation$). The Wilcoxon test (at 0.05 level) shows that the null-hypothesis should be rejected (with p-value = 0.5059 for *TD* and p-value=0.1373 for *TDF*) thus the methods cannot be distinguished according to the number of rules. However, it should be noted that for all data sets except two (*auto93* for the *TD* algorithm and *lowbwt* for the *TDF* algorithm) the *Hill climbing* algorithm returns a smaller number of rules.

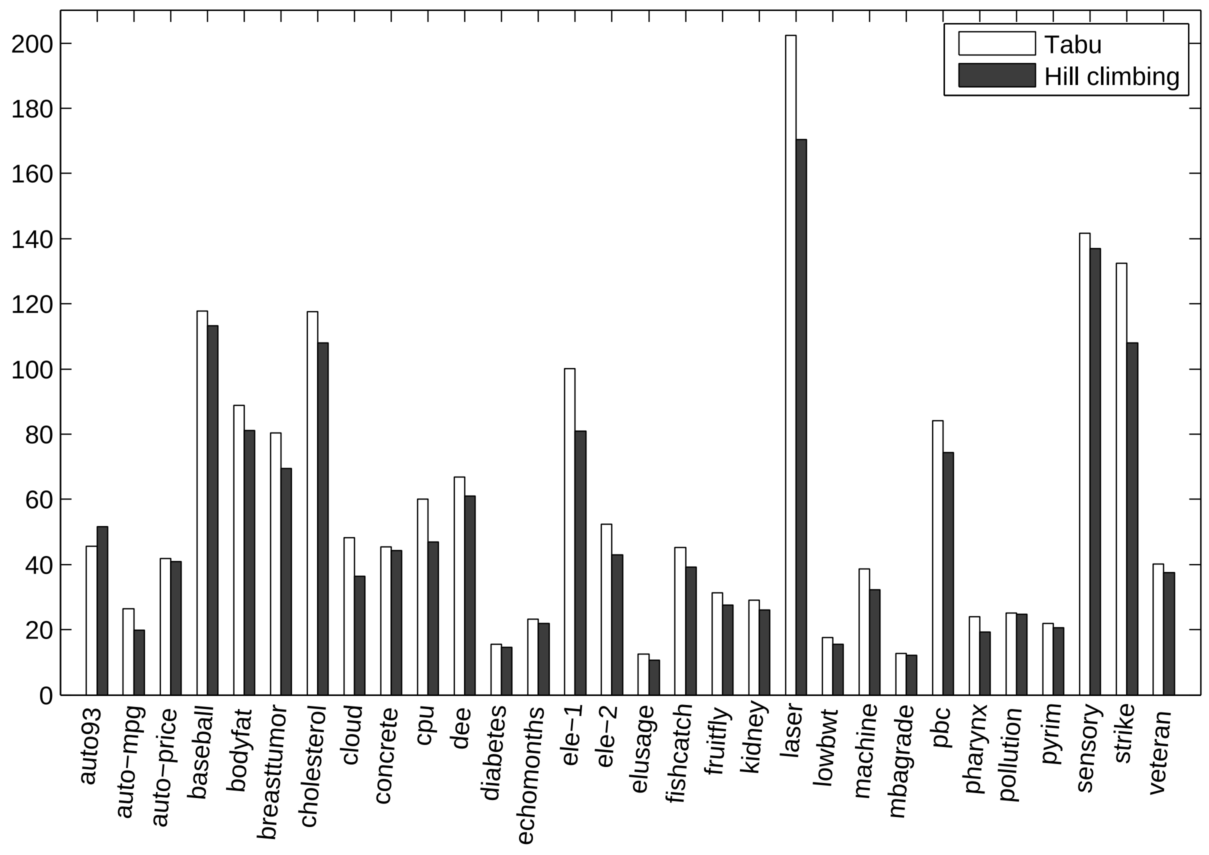


Figure 5.8: Comparison of an average number of rules after the use of the *Tabu hill climbing* and *Hill climbing* pre-pruning methods respectively in the case of the Top-Down algorithm.

The results lead to the conclusion that, generally, the *Hill climbing* approach seems to be a better choice. Although, due to the prediction error there is no reason to exclude the *Tabu hill climbing* method but taking the number of rules into account, the *Hill climbing* method seems to be the better choice, because of the simpler form of the model. One can also conclude that the extension of the simplest approach did not decrease the prediction error and did not reduce the size of the model. Therefore, it is recommended to use the *Hill climbing* pre-pruning algorithm as a quick and simple method.

5.6. Mixing measures for growing and pruning phase

Mixing measures for the growing and pruning process is a consequence of research on the pre-pruning methods. The motivation for such an approach had been presented in the previous section (see Section 5.5). To recall it at a glance, the separation was proposed to ensure that both phases may have completely different objectives. Thus, it seems natural to examine this direction in relation to the regression rules.

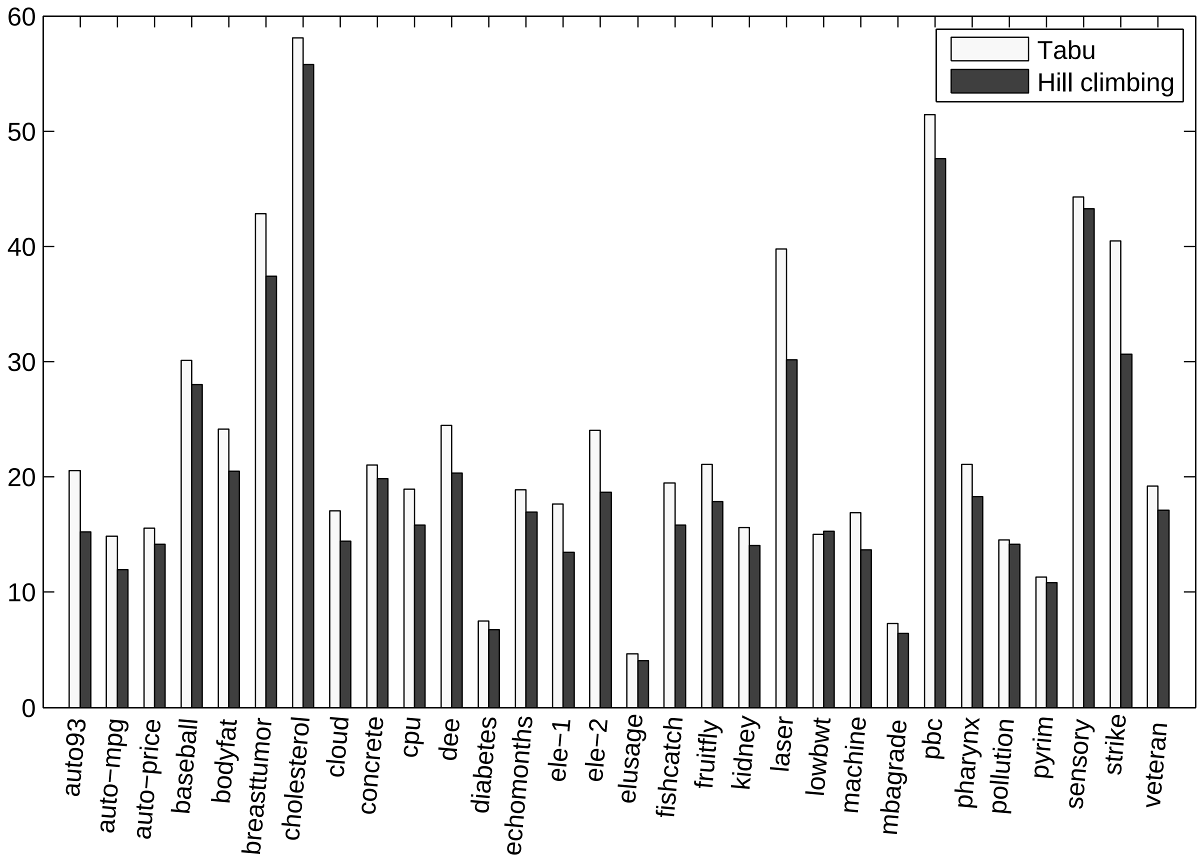


Figure 5.9: Comparison of an average number of rules after the use of the *Tabu hill climbing* and *Hill climbing* pre-pruning methods respectively in the case of the Top-Down Fixed algorithm.

Table 5.18: Average RRSE for all algorithms and selected quality measures on 30 benchmark data sets.

	TD	TDF	BU	BUF	Wilcox
C2	78.50	75.26	76.35	78.91	-
C2\Correlation	78.68	76.05	76.03	78.01	0.9234
Correlation	83.89	72.50	77.44	77.57	-
Correlation\C2	84.41	73.69	78.57	82.75	0.7845

Table 5.18 shows average RRSE of algorithms for chosen heuristics. The Friedman test indicates significant differences (F_F = 3.756 with respect to the critical value F(4,30) = 2.6896, p-value = 0.016) between the compared heuristics and shows that the selection of the heuristic has a impact on the prediction error. The comparison of the pairs of measurement due to the measure, which was used in the growing phase of rules induction, revealed that they are no significant differences between the pairs *C2-C2\Correlation* and *Correlation-Correlation\C2* (p-values are presented in the last column of Table 5.18). The visualization of comparisons is shown in Figure 5.10. The heuristics that are not significantly different according to the

Nemenyi test (at 0.05 level) are connected. The visualization shows that there is a statistically significant difference only between heuristics with different quality measures for the growing and pruning phase. This diagram also shows, despite the lack of differences, that the pair *C2\Correlation* has the highest average ranks, while the pair *Correlation\C2* is characterized with the lowest ones. However, both pairs do not, differ from the standard quality measures.

Table 5.19: Average number of rules for all algorithms and selected quality measures on 30 benchmark data sets.

	TD	TDF	BU	BUF
C2	72.30	30.08	19.21	17.45
C2\Correlation	36.56	13.36	13.94	13.35
	-49.42%	**-55.59%**	-27.47%	-23.50%
Correlation	44.85	15.45	10.70	7.19
Correlation\C2	63.55	27.43	12.37	7.93
	$+41.71\%$	**+77.49%**	$+15.65\%$	$+10.25\%$

Table 5.19 presents an average number of rules induced for each algorithm and the percentage change in the number of rules on the use of a pair of quality measures in relation to the number of rules generated using the same measure in the growing and pruning phase (calculated as $100\% \cdot (1 - n_1/n_2)$, where n_1 is the number of rules for the heuristic with the same measure for both phases and n_2 is the number of rules with mixtures of quality measures). It can be noticed that for each algorithm the pair *C2\Correlation* allows to reduce a rule set by 23%-56% without affecting much its predictive abilities. On the other hand, the pair *Correlation\C2* contributes to increase the number of rules by 10%-78%. It can also be observed that the reduction/increase is greater for the TD and TDF algorithms, which generally return more rules, than for the BU and BUF algorithms.

The results showed that the use of different quality measures for the growing and pruning phase in the regression rule induction process may lead to different result in both RRSE and the number of rules. The application of the *Correlation* quality measure (which, as shown also in Section 5.3, has a tendency to return a smaller number of rules) in the pruning phase allowed to maintain a similar prediction error like the one obtained by the *C2* quality measure. Moreover, this experiment proved that the application of the *Correlation* measure in the pruning phase, in combination with the *C2* quality measure in the growing phase *C2*, allows to reduce the number of rules by 23%-56%, which significantly improves the readability of the model. This impels us to examine this pair of quality measures in further experiments.

By contrast, this experiment also suggests that the application of the quality measure, that tends to return a larger rule-based model, does not improve the prediction error, but considering the bigger number of rules could be a good choice for descriptive purposes.

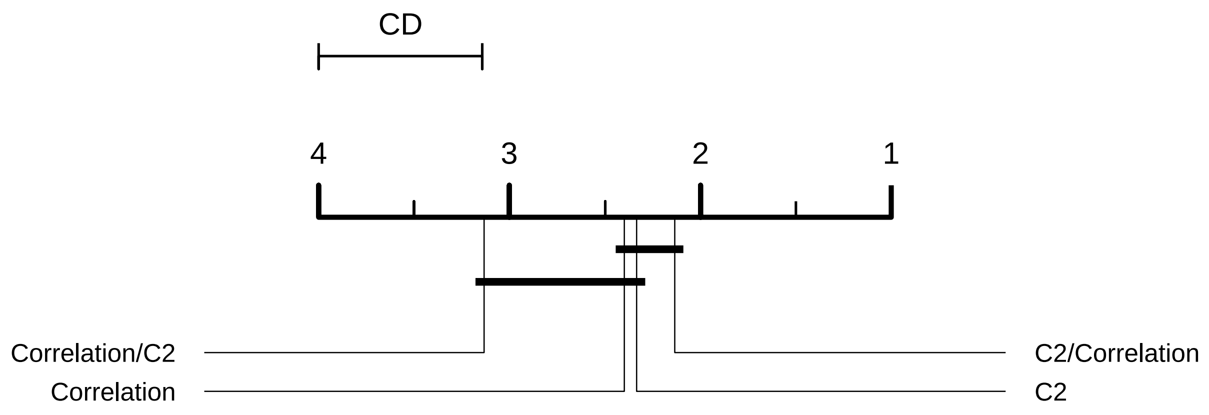


Figure 5.10: Comparison of quality measures against each other with the Nemenyi test at $\alpha = 0.05$ in the mixing measures experiment.

5.7. Conflict resolution problem

The evaluation of conflicts resolution approaches has been performed based on the results of all four rule induction algorithms (TD, TDF, BU, BUF) and three heuristics ($C2$, $C2$), ($Correlation$, $Correlation$) and ($C2$, $Correlation$) on 30 data sets. The results were averaged for each algorithm. These results were then compared.

Table 5.20: Comparative table of the results of statistical tests against the differences between 4 methods for solving conflicts and 3 pairs of quality measures over 30 data sets for each method. The critical value for the Friedman test is 2.6896. The plus sign next to the value of the Friedman test indicates the significant difference between methods in the given algorithm.

algorithm	Friedman test	p-value
TD	15.24^{+}	$8.5005 \cdot 10^{-7}$
TDF	101.24^{+}	$4.3299 \cdot 10^{-15}$
BU	28.74^{+}	$1.0207 \cdot 10^{-9}$
BUF	61.78^{+}	$3.1808 \cdot 10^{-13}$

The Friedman tests show statistical significant difference between conflicts resolution approaches in each of the proposed algorithms 5.20. It means that the choice of the appropriate method of conflict resolution has a huge impact on the prediction error of the obtained rule-based data model. The differences in each algorithm were visualized by the Nemenyi post-hoc test at 0.05 significance level.

Figure 5.11 for the *Top-Down* rule induction algorithm indicates 3 groups of conflict resolution methods between which it has not been possible to notice the statistically significant differences. The best result was obtained for the *mean of coverage* method but it is not possible to say that this method is statistically better than the *intersection of coverage* method. The *intersection of coverage* method is, however, also indistinguishable to the *median of coverage* method. The worst method is *max rule quality* and there is statistical evidence that this method gives results inferior to the methods *mean of coverage* and *intersection of coverage*. However, referring to the ranks it is clearly visible that the *mean of coverage* and *intersection of coverage* methods are much better than the other two.

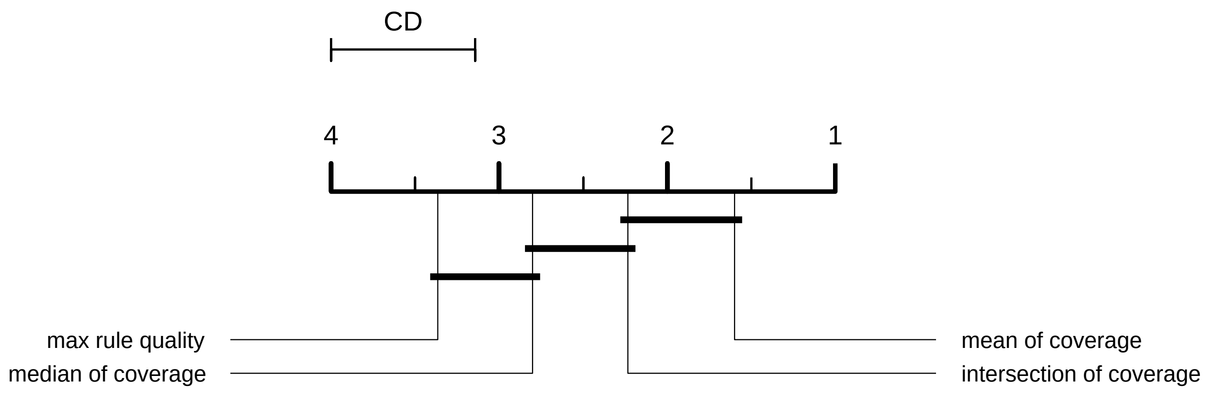


Figure 5.11: Comparison of all 4 conflict resolution methods against each other with the Nemenyi test for the *TD* algorithm. Groups of methods that are not significantly different (at $p = 0.05$) are connected.

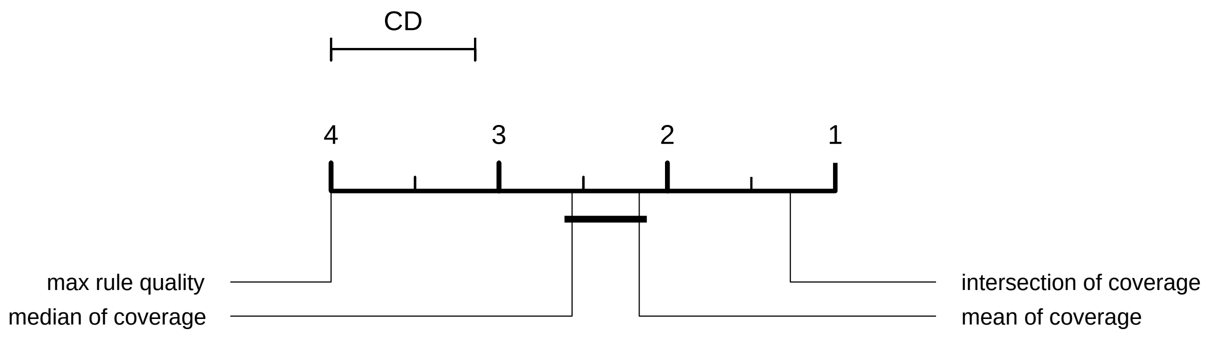


Figure 5.12: Comparison of all 4 conflict resolution methods against each other with the Nemenyi test for the *TDF* algorithm. Groups of methods that are not significantly different (at $p = 0.05$) are connected.

Figure 5.12 presents the result for the *Top-Down Fixed* algorithm which is the most interesting result with respect to all the four algorithms. The Nemenyi test for the *Top-Down Fixed* algorithm allowed to identify only 1 pair of methods with the lack of statistically significant difference. These are methods: *mean of coverage* and *median of coverage*. The *intersection of coverage* method, however, outperforms other methods, which is heavily accented by the position (ranking) of this method in Diagram 5.12. In turn, the worst method is *max rule quality* and statistically this is not the preferred method for *Top-Down Fixed* algorithm.

Figure 5.13 shows the results obtained for the *Bottom-Up* algorithm. The order of methods for conflict resolution is the same as for the *Top-Down* algorithm, but this time there is only

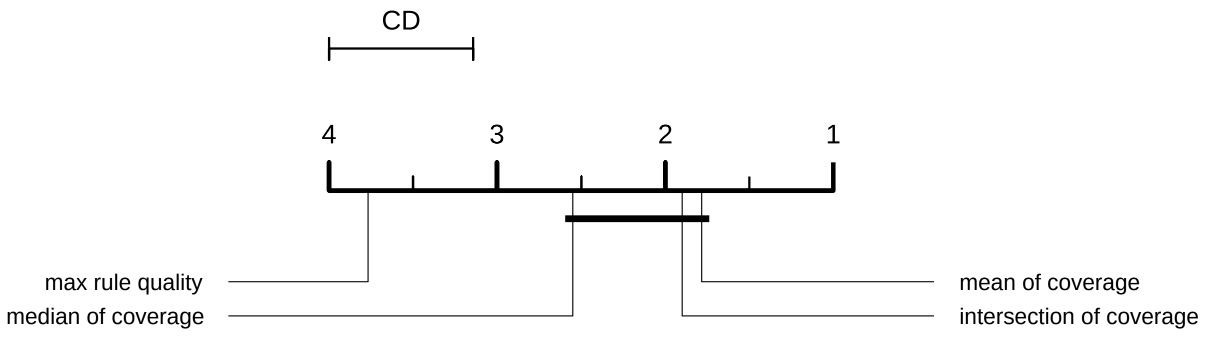


Figure 5.13: Comparison of all 4 conflict resolution methods against each other with the Nemenyi test for the *BU* algorithm. Groups of methods that are not significantly different (at $p = 0.05$) are connected.

one group of of indistinguishable methods. The only method that stands out from the rest is the worst *max rule quality* method. It is also worth noting that the methods *mean of coverage* and *intersection of coverage* allow to obtain very similar results (they are close to each other on the CD diagram).

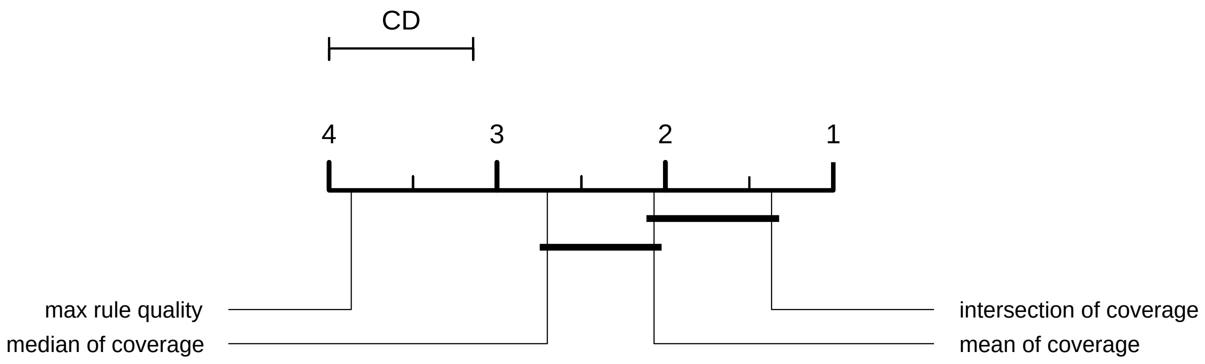


Figure 5.14: Comparison of all 4 conflict resolution methods against each other with the Nemenyi test for the *BUF* algorithm. Groups of methods that are not significantly different (at $p = 0.05$) are connected.

The Nemenyi test for the *Bottom-Up Fixed* algorithm allowed to identify 2 groups of indistinguishable methods (see Figure 5.14). The first group consists of the *intersection of coverage* and *mean of coverage* method while the second one consists of the *mean of coverage* and *median of coverage* method. Based on the obtained ranks the order of methods is similar to the *Top-Down Fixed* algorithm. The best method proved to be *intersection of coverage* and the worst *max rule quality*.

While analyzing the results, a few patterns can be seen. Firstly, in all cases the worst method proved to be *max rule quality*. In the case of standard approaches of the *Top-Down* and *Bottom-up* algorithms the best method was *mean of coverage*, even if this method was not statistically better than the second one, *intersection of coverage*. Moreover, for the *Fixed* version of both algorithms it is highly visible that the order is reversed and for the *Top-Down* algorithm the *intersection of coverage* method outperforms other methods. Although in the case of the *Bottom-Up Fixed* algorithm the Nemenyi test did not confirm the superiority of the *intersection of coverage* method over the *mean of coverage* one, the distance between the methods is very close to critical and thus there is strong evidence for the *intersection of coverage* method.

It is an interesting fact that the *max rule quality* method, which predicts the target value for test examples based on the rule with the highest quality on training data set, leads to the worst results. The possible hypothesis of such a result could be that one rule is insufficient for determining accurate prediction due to other attributes, invisible to this rule, that can be included in other rules and thus may affect the accuracy of prediction. The different rules then combine predictive abilities to more accurate estimation of the target value.

It seems that for the standard approaches of both rule induction algorithms, *mean of conclusion* is a preferable method to resolve conflicts. It is the simplest and the fastest method to predict the target value, and, as shown in the experiments, it leads to the lowest prediction error. Therefore, this is the recommended method of conflict resolution for this type of approach. In the case of *Fixed* version it was proven that the *intersection of coverage* method allows for more accurate prediction, and in the case of the *Top-Down Fixed* algorithm it also statistically outperforms other methods. Then it seems reasonable to recommend the *intersection of coverage* method to resolve conflicts.

5.8. Post-pruning methods evaluation

The research on the algorithms for filtration of unordered sets of regression rules has been performed for six post-pruning algorithms *Inclusion*, *Coverage*, *Disjoint*, *Forward*, *Backward*, and *Forward-Backward*. The results are presented separately for each rule induction algorithm (TD, TDF, BU and BUF), for three selected pairs of quality measures for the growing and pruning phase respectively $(C2, C2)$, $(Correlation, Correlation)$ and $(C2, Correlation)$. This approach allows to draw conclusions independently of the applied quality measure or the used induction algorithm and it should help to choose one universal post-pruning algorithm that could be used in any experiment or the induction algorithm.

Tables 5.21, 5.22 and 5.23 present average results of filtering algorithms for the chosen heuristic. All values from those tables (except the p-values in the fourth column) are averages over 30 data sets. The successive columns (starting from the left) are: the name of the algorithm with the chosen heuristic used for the rule induction for both growing and pruning phase; the name of the rule filtering algorithm (the rows marked as *None* correspond to the rule set without filtration); RRSE, the p-value of the Wilcoxon signed-rank test (rounded to six significant digits); the number of rules in the final set; the percentage of the reduction in the number of rules (calculated as $100\% \cdot (1 - n_1/n_2)$, where n_1 and n_2 are the number of rules after and before filtration respectively); percentage of test examples which were covered by the default rule. For the Wilcoxon test and in the calculation of the reduction, the rule sets before filtration were treated as the reference. Additionally, the p-values smaller than 0.05 are marked with the sign $^+$ and $^-$ which show a statistically significant (at 0.05 level) improvement / degradation of the prediction error over the prediction error of rule sets without filtration.

Table 5.21: Average result of the rule induction and filtering algorithms for the C2 measure.

algorithm	filtration	rrse	wilcox	#rules	reduction (%)	default (%)
TD C2	None	79.34	–	74.71	–	1.78
	Inclusion	79.58	0.135908	65.65	12.12	**1.80**
	Coverage	79.42	0.991795	51.55	31.00	2.16
	Disjoint-0.9	80.27	0.120445	40.41	45.91	9.98
	Backward	74.53	$0.000771^{(+)}$	32.83	56.06	9.61
	Forward	**74.18**	$0.001197^{(+)}$	29.40	60.65	20.15
	ForwBack	74.41	$0.004682^{(+)}$	**23.19**	**68.96**	23.01
TDF C2	None	79.98	–	31.08	–	1.54
	Inclusion	80.73	$0.025637^{(-)}$	27.82	10.51	**1.59**
	Coverage	80.46	0.130592	23.87	23.21	1.94
	Disjoint-0.9	81.10	0.289477	**17.54**	**43.57**	7.79
	Backward	**76.12**	$0.000148^{(+)}$	21.96	29.35	6.12
	Forward	76.59	$0.005320^{(+)}$	19.69	36.64	13.71
	ForwBack	76.70	$0.010444^{(+)}$	17.82	42.67	15.82
BU C2	None	79.23	–	19.85	–	8.71
	Inclusion	79.33	$0.025641^{(-)}$	18.30	7.84	**8.74**
	Coverage	79.24	0.648801	16.44	17.19	9.05
	Disjoint-0.9	80.01	0.055664	13.95	29.75	12.91
	Backward	78.87	0.228880	13.84	30.29	20.60
	Forward	**78.32**	0.075213	14.08	29.10	22.25
	ForwBack	78.42	0.147040	**12.74**	**35.81**	23.64
BUF C2	None	82.34	–	18.03	–	8.37
	Inclusion	82.52	$0.045082^{(-)}$	16.60	7.95	**8.38**
	Coverage	82.90	$0.005834^{(-)}$	15.13	16.10	8.57
	Disjoint-0.9	85.10	$0.001137^{(-)}$	12.54	30.48	12.82
	Backward	80.86	$0.016878^{(+)}$	14.44	19.93	13.92
	Forward	**80.02**	0.183575	12.68	29.69	22.72
	ForwBack	80.09	0.183575	**11.99**	**33.49**	23.88

As it can been seen in all three tables, the application of filtering algorithms always returns a smaller number of rules. The reduction, however, varies according to the filtering

Table 5.22: Average result of the rule induction and filtering algorithms for the Correlation measure.

algorithm	filtration	rrse	wilcox	#rules	reduction (%)	default (%)
TD Corr	None	86.33	–	46.34	–	0.89
	Inclusion	86.31	0.428430	38.59	16.74	**0.89**
	Coverage	86.11	0.360039	33.28	28.20	0.94
	Disjoint-0.9	86.52	0.585712	24.61	46.90	7.86
	Backward	81.25	$0.000189^{(+)}$	20.31	56.18	10.89
	Forward	79.90	$0.001036^{(+)}$	15.79	65.92	24.16
	ForwBack	**79.57**	$0.000490^{(+)}$	**12.95**	**72.06**	26.60
TDF Corr	None	79.83	–	15.97	–	1.00
	Inclusion	80.77	$0.000174^{(-)}$	12.25	23.28	**1.08**
	Coverage	81.95	$0.000529^{(-)}$	10.61	33.57	1.31
	Disjoint-0.9	84.50	$0.002765^{(-)}$	**7.11**	**55.45**	6.63
	Backward	76.18	$0.002585^{(+)}$	12.79	19.92	3.98
	Forward	**75.10**	$0.000963^{(+)}$	11.12	30.38	12.26
	ForwBack	75.13	$0.002105^{(+)}$	10.50	34.22	13.43
BU Corr	None	81.59	–	11.05	–	5.31
	Inclusion	81.74	0.057533	9.68	12.39	**5.37**
	Coverage	81.58	0.584713	8.85	19.96	5.53
	Disjoint-0.9	81.04	0.289477	7.27	34.26	8.54
	Backward	79.53	$0.004114^{(+)}$	7.50	32.15	21.39
	Forward	79.42	$0.009271^{(+)}$	7.67	30.61	20.26
	ForwBack	**79.40**	$0.007271^{(+)}$	**7.06**	**36.10**	23.22
BUF Corr	None	83.57	–	7.43	–	6.15
	Inclusion	84.14	$0.022541^{(-)}$	6.54	12.02	**6.16**
	Coverage	84.36	$0.004940^{(-)}$	6.03	18.89	6.24
	Disjoint-0.9	87.03	$0.002649^{(-)}$	**5.09**	**31.49**	8.67
	Backward	**80.00**	$0.015909^{(+)}$	6.33	14.76	14.78
	Forward	81.71	0.399484	5.59	24.81	22.62
	ForwBack	81.32	0.346905	5.38	27.59	25.89

Table 5.23: Average result of the rule induction and filtering algorithms for the C2 measure used in the growing phase and Correlation used in pruning phase.

algorithm	filtration	rrse	wilcox	#rules	reduction (%)	default (%)
TD C2/Corr	None	79.75	–	37.78	–	0.89
	Inclusion	79.96	0.673280	27.60	26.94	**0.90**
	Coverage	79.85	0.530440	22.78	39.70	1.07
	Disjoint-0.9	81.68	0.065641	17.59	53.45	6.42
	Backward	74.64	$0.000205^{(+)}$	17.25	54.34	12.61
	Forward	74.98	$0.003379^{(+)}$	14.81	60.80	21.67
	ForwBack	**74.55**	$0.000894^{(+)}$	**12.51**	**66.90**	24.39
TDF C2/Corr	None	81.37	–	13.80	–	1.10
	Inclusion	81.67	0.205888	11.05	19.95	**1.11**
	Coverage	83.24	$0.008217^{(-)}$	9.29	32.72	1.34
	Disjoint-0.9	86.20	$0.001709^{(-)}$	**6.48**	**53.03**	5.54
	Backward	77.58	$0.000136^{(+)}$	10.84	21.49	5.93
	Forward	**77.45**	$0.006035^{(+)}$	9.30	32.60	15.42
	ForwBack	77.50	$0.007271^{(+)}$	8.76	36.56	17.04
BU C2/Corr	None	78.97	–	14.4	–	6.03
	Inclusion	79.48	$0.001285^{(-)}$	12.27	14.81	**6.08**
	Coverage	79.72	$0.007271^{(-)}$	10.84	24.72	6.31
	Disjoint-0.9	81.04	$0.001114^{(-)}$	**8.73**	**39.35**	9.57
	Backward	**78.51**	0.236936	10.64	26.09	18.72
	Forward	79.11	0.845080	10.65	26.06	18.93
	ForwBack	79.09	0.942611	9.90	31.25	21.08
BUF C2/Corr	None	80.01	–	13.35	–	6.75
	Inclusion	80.77	$0.000247^{(-)}$	10.62	20.42	**6.80**
	Coverage	81.40	$0.000016^{(-)}$	9.52	28.71	6.90
	Disjoint-0.9	84.45	$0.000021^{(-)}$	**7.64**	**42.75**	9.88
	Backward	**78.50**	0.369525	11.21	16.00	12.69
	Forward	79.13	$0.045484^{(+)}$	10.29	22.95	17.09
	ForwBack	79.13	0.381172	9.78	26.72	19.68

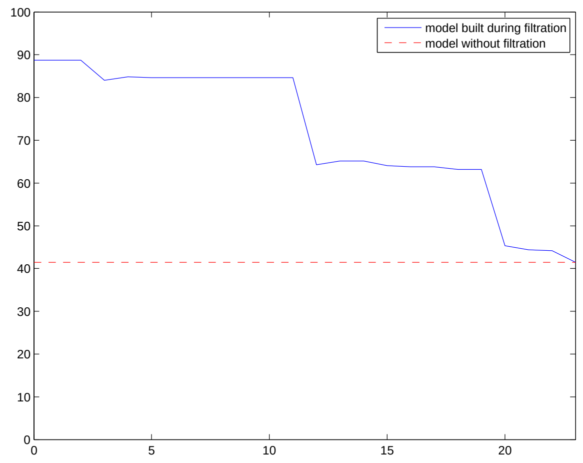
algorithm. The lowest reduction was observed for the *Inclusion* algorithm (except one case for the *BUF* algorithm and $(C2, Correlation)$ heuristic), and the highest for *Disjoint* and *ForwBack*. However, irrespective of the used heuristics, the best reduction and the lowest prediction error were achieved at the cost of the higher value of percentage of test examples that were covered by the default rule.

According to the results of the Wilcoxon test, in all cases the first three filtering algorithms (*Inclusion*, *Coverage* and *Disjoint*) are statistically indistinguishable compared to *None*, or cause a statistically significant increase in the value of the prediction error, while for the group of the last three algorithms the value of the prediction error is statistically indistinguishable or significantly improves the prediction.

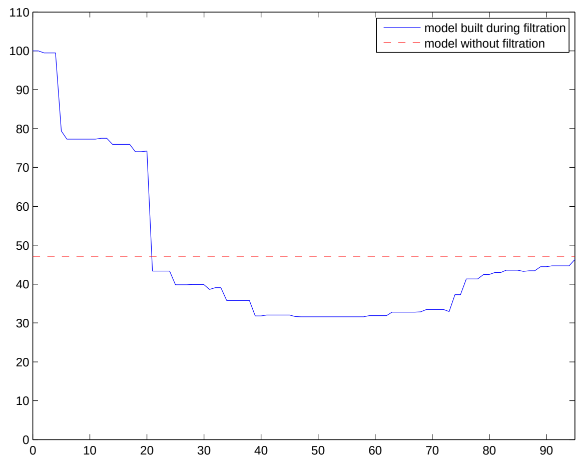
Comparing algorithms with each other, the *Inclusion* algorithm behaves similarly to *Coverage*. However, the *Coverage* algorithm is characterized by a higher reduction rate, and therefore it is a better choice. Moreover, for the *Inclusion* algorithm the Wilcoxon signed-rank test at 0.05 level shows significant increase in the value of RRSE in 7 out of 12 experiments while the *Coverage* algorithm shows differences in one experiment less. The number of unrecognized examples is also at the similar level in each algorithm. These results incline to recommend of the *Coverage* algorithm when we want to reduce a rule set by 10%-40% without affecting much its predictive and descriptive abilities.

The most noticeable improvement in the value of RRSE was observed for the *Forward*, *Backward* and *Forward-Backward* algorithms. Especially good results were obtain by the *Backward* algorithm which in few cases improves significantly RRSE without a large increase in the number of examples covered by the default rule. While comparing these algorithms to the *Inclusion* and *Coverage* algorithms it can be seen that these three algorithms are characterized by a higher reduction rate, but the number of unrecognized test examples is also higher. Such an increase in the number of unrecognized test examples is probably caused by the fact that these filtering algorithms eliminate rules which do not improve prediction accuracy in relation to the accuracy achieved by the default rule. Generally, if the increase in the number of uncovered test examples does not play a major role in the specified task, this group of algorithms outperforms other algorithms and it is recommended. According to the Nemenyi test, there are no significant differences in RRSE at 0.05 level between the *Forward*, *Backward* and *Forward-Backward* algorithms. Therefore the choice between these three algorithms can be based on the compromise between the reduction rate and the number of unrecognized test examples.

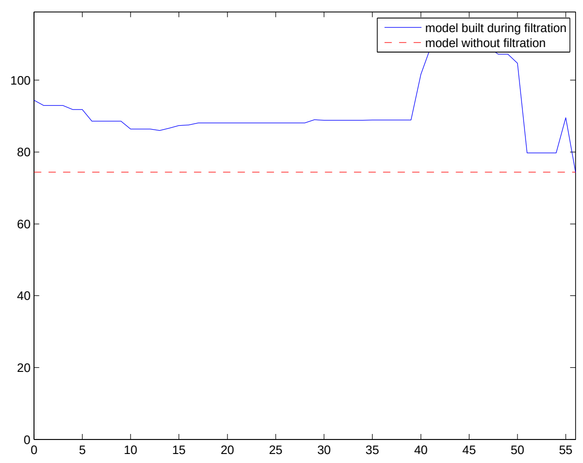
The last *Disjoint* algorithm is characterized by a very high reduction rate (in 6 of 12 cases the highest one) however at the expense of a higher prediction error. However, the Wilcoxon test indicates that for half of experiments there is no difference between the *Disjoint* algorithm



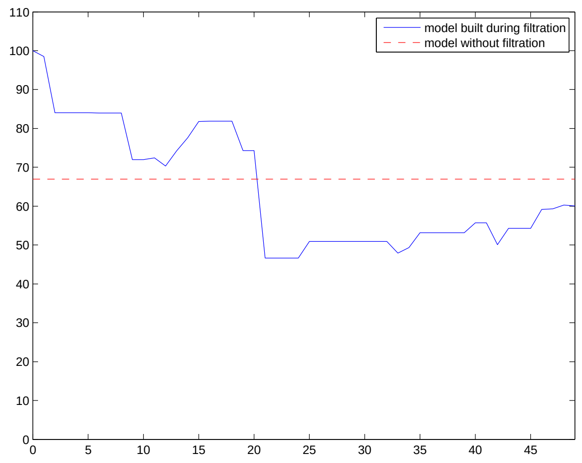
(a) autoprice



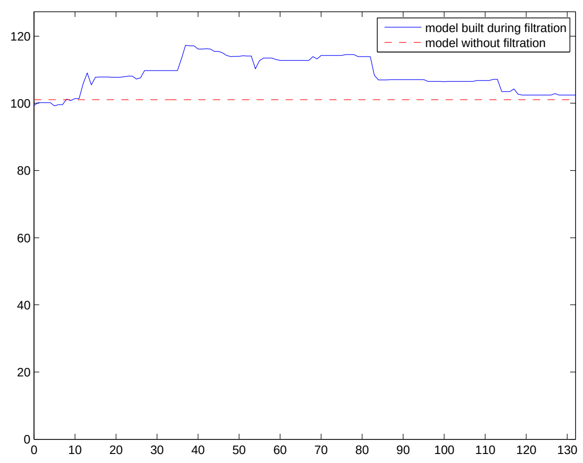
(b) bodyfat



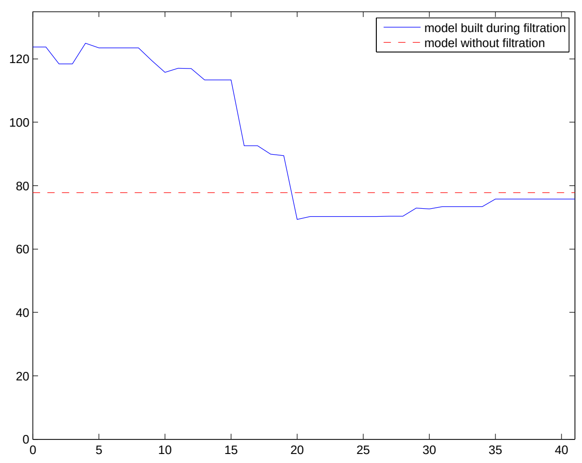
(c) cpu



(d) concrete



(e) cholesterol



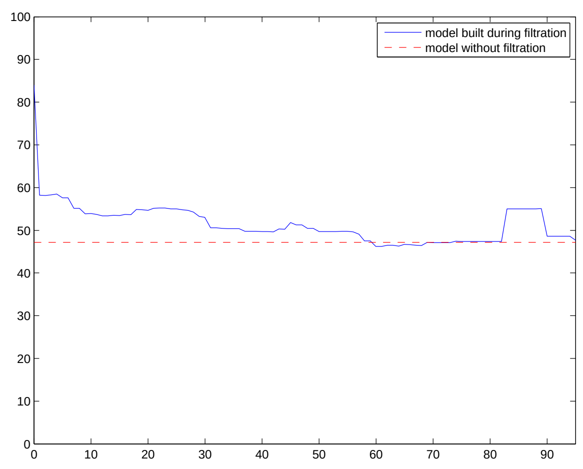
(f) pharynx

Figure 5.15: Graph of the value of prediction error (RRSE) in subsequent steps of filtration for the ruleset obtained using TD algorithm. All diagrams have been obtained for the *Coverage* filtering method and every first fold of the 10-fold stratified cross-validation for listed data sets: *autoprice*, *bodyfat*, *cpu*, *concrete*, *cholesterol* and *pharynx*. The dashed line represents the prediction error of the models without filtration.

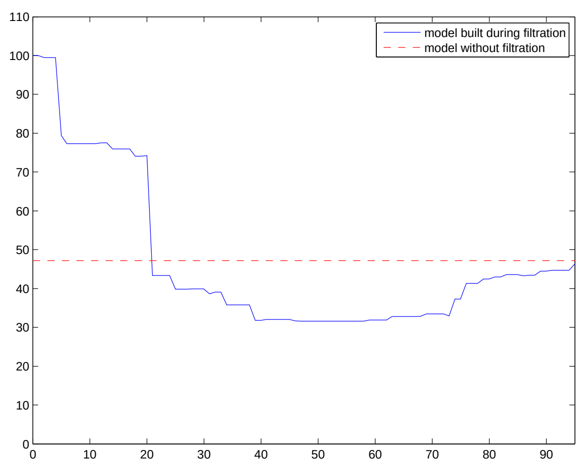
and *None*. Moreover, despite high reduction in the number of rules, the *Disjoint* algorithm still generalizes well to an independent test set. Although it has a higher percentage of unrecognized test examples (i.e. asigned to the default rule) than *Inclusion* or *Coverage*, it has a much lower percentage than *Backward*, *Forward* and *ForwBack*. In view of the reduction and generalization rate, the *Disjoint* algorithm seems to be a good choice to optimize the number of rules for descriptive purposes.

Further analysis was performed to examine the process of building the filtered rule-based model. The first experiment was conducted on the six chosen data sets (autoprice, bodyfat, cpu, concrete, cholesterol and pharynx which enable to obtain the most different results) and the *Coverage* filtering algorithm. The main aim of this experiment was to investigate the prediction error of the rule-based model in subsequent iterations of one filtering algorithm, but on different data sets. The results of this research are presented in Figure 5.15. The solid line represents the measured accuracy while the dashed line stands for the model without filtration. As it can been seen, the value of RRSE for the rule-based model with one rule is around 100% in all cases. It means that an empty model is insufficient to accurately predict the target value of the entire test set. However, interesting results can be seen on the subgraphs *a* and *c* where the addition of the first rule to the filtered model allows to obtain a lower prediction error than a simple mean of the target value of all training examples. Further iterations always cause a change in the value of the prediction error. For the presented data sets the error of prediction in subsequent iterations mostly improves, as it is represented by the subgraphs *a*, *b*, *d* and *f*. However, the error increases at some point. The subgraph of the *cpu* data set (*c*) presents the case in which the prediction error does not decrease for a long time and even temporarily increases at some point. At the final phase of filtration the value of the prediction error of the filtered model approaches the value of the prediction error of the model without filtration. It is also worth to note that in the case *b*, *d* or *f* the error of the model in the middle of the filtration process reaches its minimum and then increases. However, this is a consequence of the assumptions of the algorithm, wherein the filtering algorithm does not interrupt its activity at the local minimum but performs the filtering process based on the uniqueness of coverage of the remaining rules. The highest frequency of a monotonicity change was observed for the *concrete* and *bodyfat* data sets thus these sets were chosen to check the characteristics of others filtering algorithms.

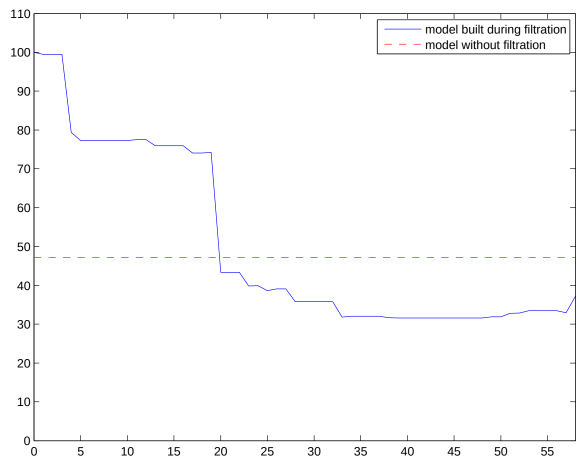
Figures 5.16 and 5.17 present the characteristics of 6 filtering algorithms for the same one fold of the 10-fold stratified cross-validation for the *bodyfat* and *concrete* data set respectively. One can observe that three algorithms (*Coverage*, *Disjoint* and *Forward*) are characterized by similar diagrams for both data sets. By analyzing the construction of these algorithms it can be concluded that one common feature for all of them is a starting point. In all algorithms the filtration process begins from the best rule which leads to the most noticeable improvement



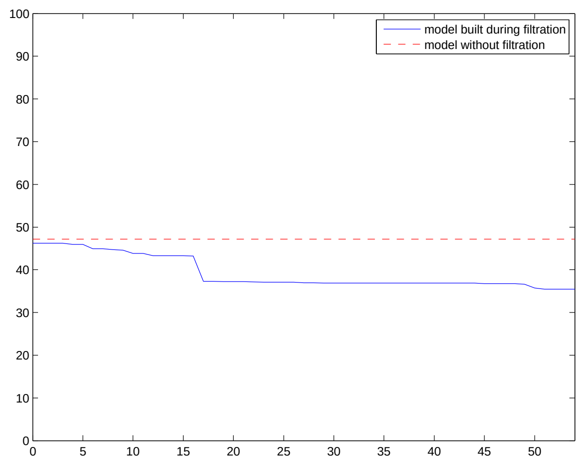
(a) Inclusion



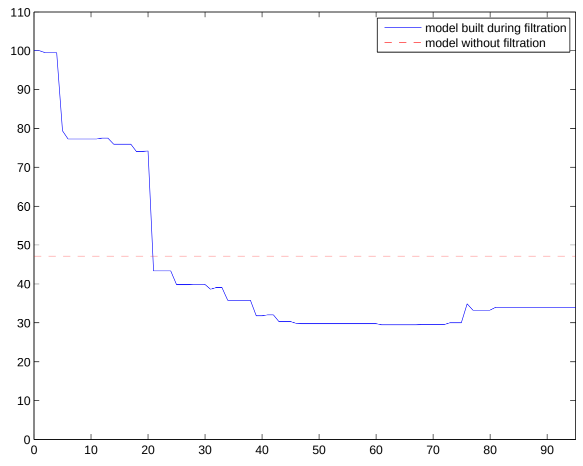
(b) Coverage



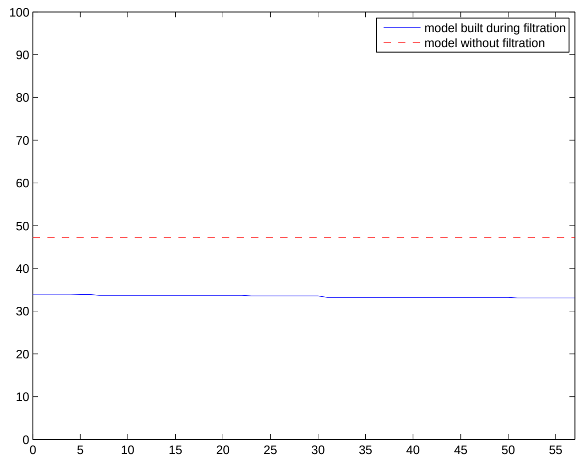
(c) Disjoint-0.9



(d) Backward

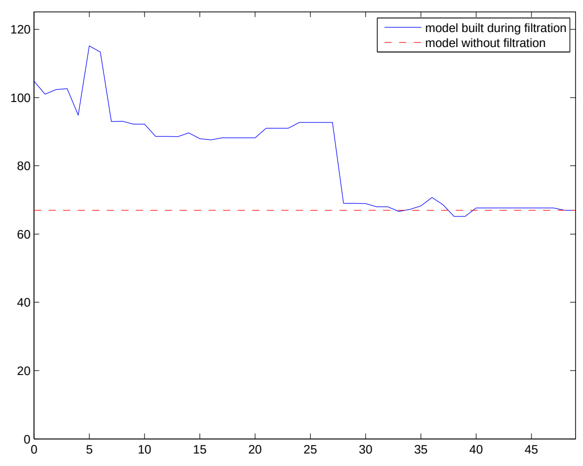


(e) Forward

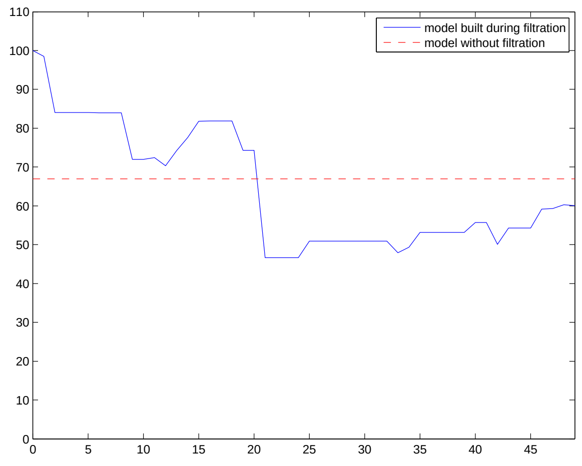


(f) ForwBack

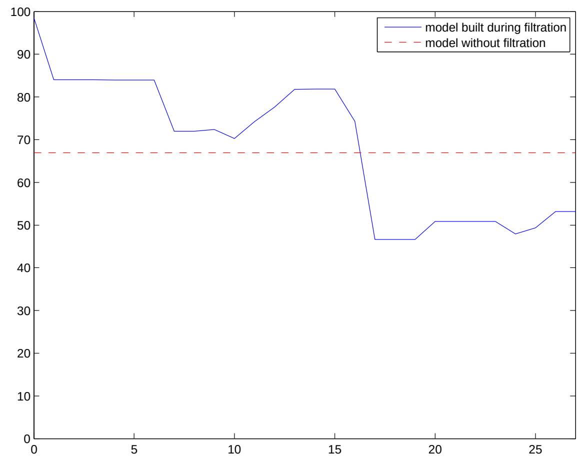
Figure 5.16: Graph of the value of the prediction error (RRSE) in subsequent steps of 6 filtering algorithms for the rule set obtained using the TD algorithm. All diagrams have been obtained for a first fold (of the 10-fold stratified cross-validation) of the *bodyfat* data set. The dashed line represents the prediction error of the models without filtration.



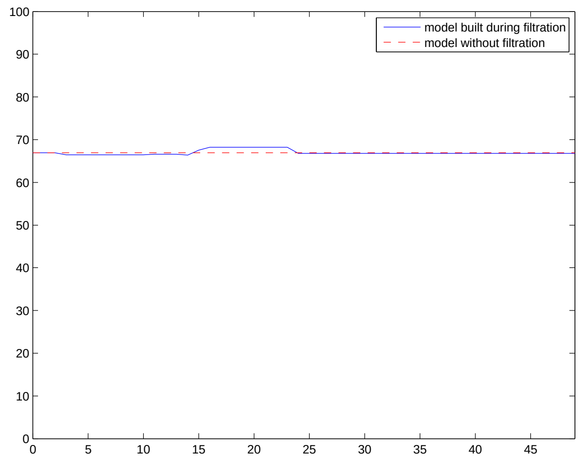
(a) Inclusion



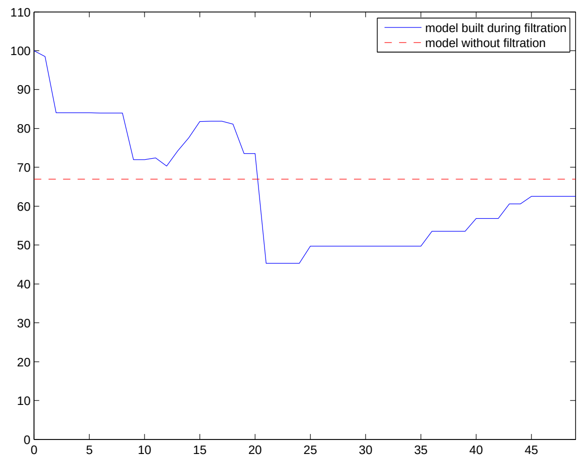
(b) Coverage



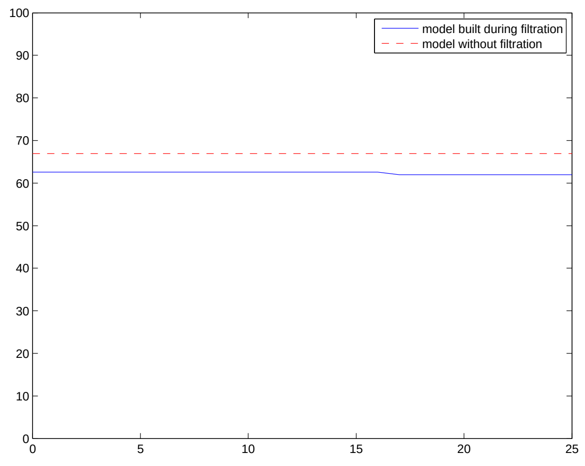
(c) Disjoint-0.9



(d) Backward



(e) Forward



(f) ForwBack

Figure 5.17: Graph of the value of the prediction error (RRSE) in subsequent steps of 6 filtering algorithms for the rule set obtained using the TD algorithm. All diagrams have been obtained for a first fold (of the 10-fold stratified cross-validation) of the *concrete* data set. The dashed line represents the prediction error of the models without filtration.

in the initial stage of filtration. The differences in the assumptions of the algorithms are reflected only in later iterations. The similar diagrams were also obtained for the *Backward* and *ForwBack* algorithms. However, in this case similar graphs show the use of the *Backward* algorithm in the unchanged form inside the implementation of the *ForwBack* algorithm.

The obtained results allow to conclude about the behaviour of each of the algorithms. The *Inclusion* algorithm focuses on the systematic improvement of the rule-based model quality which is reflected in the improvement of the model accuracy. The *Coverage*, *Disjoint* and *Forward* algorithms allow for faster and more powerful decline in the error value in the subsequent iterations. One can also notice a smaller number of iterations for the *Disjoint* algorithm with respect to other two algorithms. This is illustrated by the shift of the presented curve. The *Backward* algorithm removes unnecessary elements of the model when removing them has a positive impact on the rule set quality. In the case of the *concrete* data set, an impact on the accuracy of the model is almost invisible but for the *bodyfat* data set the accuracy is continuously but slowly improved. The last diagram for the *ForwBack* algorithm is the most static. This is due to the execution order of algorithms. The measurement of the model accuracy is performed at the end of each algorithm. In the case of *ForwBack* algorithm it is at the end of *Backward* algorithm. An input of this algorithm is, however, a model previously filtered with the use of the *Forward* algorithm, therefore, the following algorithm does not result in large changes (especially when the only difference between algorithms is an order of rules examination).

The experimental results lead to the conclusion that the application of filtering algorithms always allows to obtain a smaller number of rules and in most cases does not cause degradation in the values of RRSE, thus, they seem to be safe to use for the size reduction of regression rule sets. The performed statistical tests do not prove the existence of one universal filtering algorithm. However, the six rule filtering algorithms can be characterized by different properties which are reflected by the value of the prediction error, the reduction in the number of rules and the percentage of unrecognized test examples.

In view of all these considerations, the *Coverage* method seems to be the best choice and it is a recommended rule for filtering the algorithm for regression. An application of this method allows to reduce a regression rule set by 16%-40% without affecting much its predictive and generalization abilities. However, if the increase in the number of unrecognized test examples does not play a major role in the specified data mining task, then the *Backward*, *Forward* and *ForwBack* algorithms are preferable.

5.9. Comparison to existing methods

In this section the selected combination of proposed approaches was compared to several methods available in Weka 3.6.11 [127], in R 3.1.2 environment [98] or from other external sources. The attention has been paid to the methods that, similarly to rule-based ones, are able to express the data model in a comprehensible form. In addition, the comparison was enriched by the results of the RegENDER algorithm [24] as representative of the most computationally advanced methods of the regression rule induction. The results are also compared to the SeCoReg algorithm [34, 57, 59].

Table 5.24: Average performance of selected algorithms with respect to existing solutions.

algorithm	rrse	#rules
TD C2 Coverage	79.42	51.55
TD C2 Backward	74.53	32.83
RegENDER-50	80.92	50
RegENDER-150	76.62	150
M5Rules-R	78.55	59.03
SeCoReg M-Estimate	80.27	–
CART	77.37	–

The *M5Rules* algorithm was run with the -R parameter which causes that generated rules have a constant, instead of a linear model, in conclusion (as in the presented implementation). For *RegENDER*, the parameter of the number of output rules was set to 50 and 150. The first value corresponds to the average number of rules induced by the *Top-Down* algorithm with the *Coverage* filtration method while the second value is an arbitrary value to check the prediction error of a more complex model. For the SeCoReg algorithm the mEstimate heuristic was used as a recommended one. The rest of the parameters of algorithms were set to their default values.

The results presented in Table 5.24 show that regression rules or trees built with the use of a variety of methods with a single value in the conclusion can lead to different prediction abilities. The lowest prediction error was observed for one of the proposed combination of the *Top-Down* algorithm with the *C2* quality measure to control the induction process and with the *Backward* rule filtering method. Surprisingly, the worst result was observed for the *RegENDER-50* algorithm. Although, the Friedman test does not show statistically significant difference in the values of RRSE for the group of all 7 algorithms from Table 5.24, it should be noted that the *TD C2 Backward* algorithm leads to a much smaller number of rules than other algorithms and, at the same time, allows to obtain the lowest prediction error.

Furthermore, we decided to check whether the use of linear models allows to decrease the prediction error. For this purpose, the linear models for algorithms that have obtained the best

Table 5.25: Average performance of selected algorithms with linear models with respect to existing solutions.

algorithm	rrse	#rules
TD C2 Coverage Linear	63.82	51.55
TD C2 Backward Linear	67.36	32.83
M5P	73.27	–
Cubist	68.08	–

results for a single value in the conclusions have been applied. Then, the results were compared with two other algorithms that allow to produce linear models. The results of this experiment are presented in Table 5.25. One can notice that, in fact, the linear models help to reduce the prediction error. The lowest RRSE was obtained for the TD C2 Coverage method with linear models. It can be seen that in the case of linear models the value of the prediction error decreased from 7 to 13% (or $10 - 20\%$ if calculated as a ratio) in relation to the same methods but with a single value in the conclusion.

The Friedman statistic gave 10.68 (p-value 0.0136) which exceeded the critical value 2.6896 for confidence level 0.05. Thus, the null hypothesis that all algorithms performed equally well can be rejected. The post-hoc Nemenyi test allowed to identify 2 groups of indistinguishable methods (see Figure 5.18). The first group consists of all methods except M5P while the second one does not contain TD C2 Coverage Linear algorithm. This is equivalent to the fact that these methods differ from each other. Although the p-value is near 0.01 what means that the difference between methods is not strong enough, but it can be clearly seen that the distance between methods on the Figure 5.18 is considerable, thus the TD C2 Coverage Linear method is recommended to solve regression problem using methods with linear models in the conclusion.

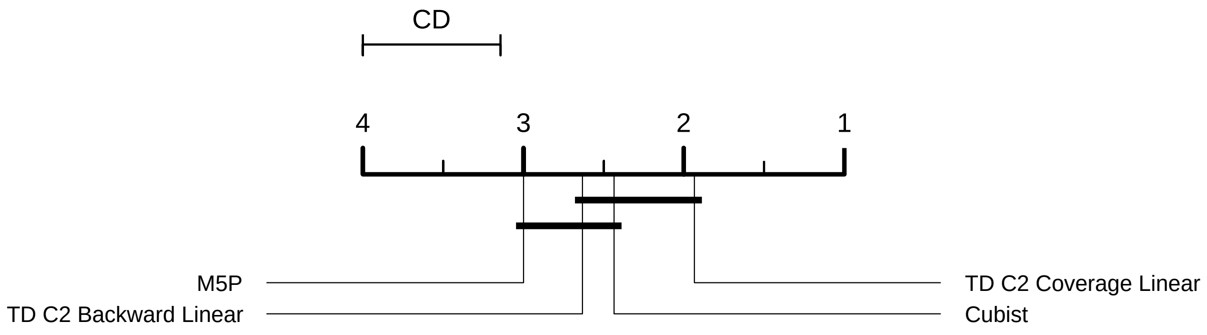


Figure 5.18: Comparison of 4 algorithms with linear models in the conclusion against each other with the Nemenyi test. Groups of methods that are not significantly different (at $p = 0.05$) are connected.

This experiment allowed to compare the proposed methods to other existing algorithms. The obtained results lead to the conclusion that the proposed methods are comparatively accurate or in some respect even better. The best results were obtained for the Top-Down algorithm with the C2 quality measure to control the induction process. The most accurate prediction was obtained for the simple mean of conclusion method to tackle with conflicts. Moreover, the filtration method leads to a lower number of rules preserving the predictive ability of the models. Finally, the linear model in the conclusion always leads to a lower prediction error but one must be aware that this happens at the cost of the readability of the model.

6. Experiments on real-life data

In this chapter the best combination of previously presented algorithms and methods is applied to real-world data. In fact, there were two real problems which had been attempted to solve by our rule algorithms. The first problem concerns methane concentration prediction in coal mines while the second problem resides in the seismic hazard prediction domain. Both tasks are of great scientific and commercial importance.

In the first part, the focus is put on the prediction of methane concentration in a coal mine. This problem, due to the huge impact on the health and lives of miners, is extremely important in today's mining industry. The main goal of this research was to predict the gas concentration that will be registered by a sensor in the consecutive time intervals (e.g. in the case of temporary unavailability of the sensor). The addition goals were to analyze data to find possible dependencies and to predict methane concentration with missing indications of a few sensors. The results of the work have already been published in the paper Regression Rule Learning for Methane Forecasting in Coal Mines [67].

In the second part, the focus is put on the seismic hazard, which is another threat in the mining industry. The goal of this work was to predict the sum of seismic energy of recorded tremors and energy recorded by the geophone in the consecutive time interval. The main challenge of this work was to translate the problem, mainly undertaken in the literature in the context of the classification, to the problem of regression, which allows to predict not only the state, like in the case of classification, but also the value of the tremors energy.

Both sections are organized in a similar way: they begin with a short introduction to the problem domain, then the data set is described and the goals are defined. In the third part the experimental settings are presented including the presentation of all compared algorithms. Finally, the results are presented and the conclusions are drawn.

6.1. Methane concentration prediction

The natural hazard monitoring systems have become a typical solution in the modern industry. Coal mining is a branch of industry where safety monitoring systems play a key role. The miners working underground in coal mines are exposed to injury or loss of life due to various environmental factors such as methane explosion or rock-bursts. Accordingly, the possibility to predict gas concentrations (methane in particular) that will be registered by sensors in the consecutive time intervals is desirable and can be even more important then monitoring the current state of the process parameters to ensure the safety of miners.

The prognostic models of that type are scarce and further research and development in this area is still needed. Some research on the prediction of the methane concentrations was conducted in the recent years [107, 108]. However, these approaches utilized the readings of the sensor being the subject of forecasting. In such a case the past indications of the sensor have obviously the strongest impact on the predicted future indications of this sensor, which makes it the most important attribute of the created prediction model.

The motivation of this work is fourfold. Firstly, it would be interesting to verify the possibility of methane concentration prediction in the place where the sensor is located only for a limited time (e.g. a portable sensor). Secondly, our approach will enable the prediction even if the indications of the existing sensor are missing for some reasons (the previous approaches utilizing the prior measurements of the sensor being a subject of prediction are not able to perform this task). Therefore, the results of this work can be utilized in such tasks as automatic filling of long lasting missing sensor measurements or automatic identification of the sensor measurements manipulations. Thirdly, it would be interesting to find out the dependencies in the analyzed data. Fourthly, the undertaken investigations are the component of a more extensive work, which is a decision support system to assist dispatchers and users of monitoring systems.

6.1.1. Data set

This study features an analysis of measurements collected in a coal mine. The methane concentration which is predicted can depend on a current mining activity, methane concentration in other locations and ventilation measured in several ways.

The data set containing information about gases concentrations was registered on a longwall outlet (with the highest risk of methane hazard). The topology of the coal mine part where the measurements were performed is presented in Fig. 6.1. The sensors indicated in Fig. 6.1 are explained in Table 6.1.

The measurement frequency of each sensor was 1 second. The data set contains measurements of 1 week aggregated at each 30 seconds. The aggregation functions applied to each sensor data are presented in Table 6.1. The task was to predict the maximal value of MM116 for next 3 minutes. The other sensors described in Table 6.1 collected the values of the attributes utilized in a prediction model. Additional attribute (PD) of the model identifies if the combine works at a given time (dominant was applied as an aggregation function). The characteristics of the collected data are presented in Table 6.2.

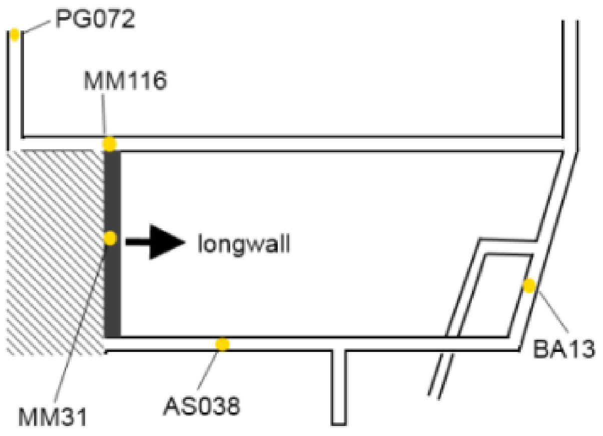


Figure 6.1: Coal mine topology and sensors.

Sensor	Sensor type	Description	Aggregation function	Variable type
MM116	methanometer	methane concentration [%]	max	dependent
MM31	methanometer	methane concentration [%]	max	independent
AS038	anemometer	air velocity [m/s]	min	independent
PG072	airflow	airflow [m^3/s]	min	independent
BA13	barometer	pressure [hPa]	mean	independent
PD	-	combain activity	dominant	independent

Table 6.1: Description of the sensors marked in Fig. 6.1.

Sensor	Min	Max	Median	Mean	Standard deviation
MM31	0.17	0.82	0.36	0.36	0.117
MM116	0.20	2.20	0.80	0.80	0.286
AS038	1.40	2.70	2.30	2.29	0.142
PG072	1.10	2.60	1.80	1.84	0.107
BA13	1067	1078	1075	1073	3.138

Table 6.2: Data characteristics.

The entire data set was divided into two disjoint parts. The first 70% of examples consisted of a training set for model building, and the last 30% of data created a test set for model evaluation.

6.1.2. Experiment and experimental settings

The *Top-Down* algorithm and *Top-Down Fixed* modification were compared with several methods available in Weka 3.6.11 [127] and in the R 3.1.2 environment [98]. The attention has been paid to the methods that, similarly to the rule-based ones, are able to express the data model in a comprehensible form. The methods that were applied are listed in Table 6.3. All algorithms were run in their default configuration, except the Cforest algorithm for which the number of trees was set to 1000.

The most important change in this experiment in relation to the other experiments of this work is the use of the Root Mean Squared Error (RMSE) measure in place of Root Relative Squared Error (RRSE). This change was made due to the fact that the experiment was only a part of a large project in which a number of methods were compared with the RMSE measure. This project was partly supported by the Polish National Centre for Research and Development (NCBiR) grant PBS2/B9/20/2013 within of Applied Research Programmes. The aim of this project was to develop a decision support system for monitoring the processes and risks in coal mines.

Id	Description	Tool
Training mean	Mean of a target attribute on a training set	R
Ctree	Regression tree with statistical cut evaluation [52]	R
Cforest	Random forest utilizing Ctree method [117]	R
Cubist	Rule-based predictive models [76]	R
M5P	M5 trees [95]	Weka
M5RULES	M5 rules [51]	Weka

Table 6.3: Reference methods utilized in the analysis.

6.1.3. Results

The results of the analysis are presented in Tables 6.4 and 6.5. The performance of each algorithm is described by the root mean squared error (RMSE), maximal error on a test set, the percentage of a number of errors above the 0.3 threshold and size. The size is calculated as the number of rules for rule-based models, the number of leaves for trees and the number of trees for random forest.

The results are divided into two tables due to the different form of the obtained model. Table 6.4 presents methods that predict a single target value while Table 6.5 presents algorithms allowing prediction with the use of the linear models. In the case of linear models, we would like to verify whether the linear models are able to reduce the prediction error. However, due to the complex structure of linear models, such models are much more difficult to interpret and thus further analysis of the descriptive ability of rules will concern only models with the simpler form of the target value.

Among the methods presented in Table 6.4 the best results were achieved by the TDF C2 filtered method. Similarly, the results before filtration (TDF C2) are characterized by a very low RMSE value, however the postprocessing reduces the number of rules significantly, which

	RMSE	Max error	% Max error over 0.3	Size
TD C2	0.265	0.773	27.09	934
TDF C2	**0.181**	**0.611**	**10.82**	**100**
TD C2 filtered	0.249	0.757	23.88	510
TDF C2 filtered	**0.177**	**0.614**	**8.59**	**18**
Training mean	0.298	0.802	47.64	-
Ctree	0.216	0.600	16.88	137
Cforest	0.206	0.566	16.27	1000
M5RULES	0.220	0.700	18.80	53

Table 6.4: Results of the analyzed methods with the single target value.

makes them intelligible and reduces the RMSE even more. Among the methods that return linear models, once again the best result were obtained by the TDF C2 filtered method but the method without filtration returned very similar results. However, it can be seen that the simple model allows to obtain a lower prediction error than the linear model. These good predictive abilities of a single value were also confirmed by a lower value of the maximum error and a lower value of the percentage of a number of errors above the 0.3 threshold.

	RMSE	Max error	% Max error over 0.3	Size
TD C2 linear	0.275	1.120	25.47	934
TDF C2 linear	**0.202**	**0.625**	**16.44**	**100**
TD C2 filtered linear	0.270	1.055	26.08	510
TDF C2 filtered linear	**0.201**	**0.625**	**16.30**	**18**
Cubist	0.216	0.944	17.91	100
M5P	0.205	0.761	15.81	195

Table 6.5: Results of the analysed methods with linear model.

The histograms presenting error distributions (difference between the real value and the predicted methane concentration) for the TD and TDF methods with the single target value are presented in Fig. 6.2. It can be noticed that a larger number of the predictions performed by means of the TD method were underestimated and the maximal value of underestimation is higher. In the case of the TDF method the overall number of under- and overestimations as well as the maximal values seem to be similar. One can conclude that the histograms show that the results of the TDF method are more balanced.

The rules generated by means of the TDF method, such as the two examples with a higher value of methane concentration presented in Table 6.6 and the two examples with a lower value of methane concentration presented in Table 6.8, can be further statistically analyzed by the calculation of confidence intervals and the pessimistic, optimistic or standard quality of the rules, as it was discussed in section 3.4. Any quality measure can be calculated this way, e.g.

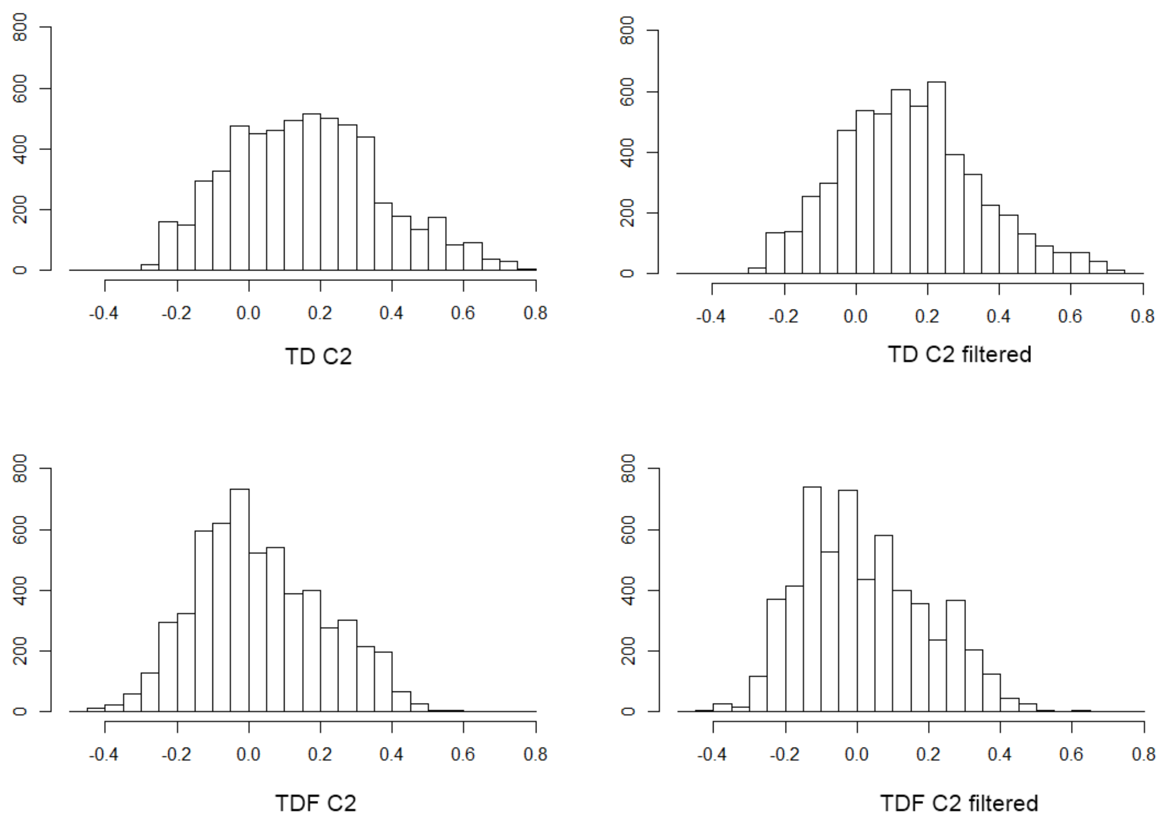


Figure 6.2: The error distributions of the *Top-Down* rule-based method.

C2 measure (2.16), precision or coverage. Tables 6.7 and 6.9 present the pessimistic, standard and optimistic values of precision and coverage quality of the rules presented in Tables 6.6 and 6.8.

	Rule R1		Rule R2
IF	PD = 1.0	IF	PD = 1.0
	BA13 $\in$ [1072.867; 1076.287)		MM31 $\geq$ 0.37
	PG072 $\geq$ 1.75		BA13 $\geq$ 1068.088
	MM31 $\in$ [0.405, 0.625)		
THEN	MM116 = 1.3 (0.200)	THEN	MM116 = 1.1 (0.201)

Table 6.6: Exemplary rules generated by means of the TDF method with higher values of methane concentration.

It can be noticed that the rule R2 is more general which results in much higher standard coverage. However, the standard precision of the rules is very similar. Looking at the ranges derived by pessimistic and optimistic values we can notice that the range of the rule R1 precision is broader than the rule R2 precision and the pessimistic precision of the rule R1 is significantly

		R1		R2	
		Precision	Coverage	Precision	Coverage
Rule quality	pessimistic	0.757	0.073	0.819	0.411
	standard	0.831	0.087	0.839	0.428
	optimistic	0.895	0.100	0.858	0.445

Table 6.7: Quality of the rules from Table 5.11.

lower then the one of the rule R2. Summarizing, the pessimistic quality (values of precision and coverage) of the rule R2 is significantly higher then the one of the rule R1.

According to the rules with lower prediction of methane concentration it can be seen that the rule R3 is characterized by higher precision and much higher coverage which results in the value of coverage. Looking at the ranges derived by pessimistic and optimistic values one can also notice that the precision range of the rule R1 is narrower than the precision range of the rule R2. The ranges for coverage are, however, very similar.

An additional knowledge about the process can be drawn out from the analysis of the structure of the rules for lower and higher prediction of methane concentration. It is evident that lower predictions were obtained where the body of the rules included such conditions like e.g.: MM31 < 0.365 for the rules R3 and R4 comparing to MM31 ≥ 0.37 for the rules R1 and R2, PD = 0 for the rule R4 and PD = 1 for the rules R1 and R2 or PG072, but in this case with intersection between ranges 1.75 and 2.05. However, looking at the minimum and maximum values for the attribute PG072 (1.10 and 2.60 respectively) it can be noticed that such left- or right-unbound intervals may still bring additional knowledge about the prediction value.

Rule R3		Rule R4	
IF	AS038 $\geq$ 2.05	IF	MM31 $<$ 0.365
	BA13 $\in$ [1070.228; 1077.683)		PD = 0
	PG072 $<$ 2.05		BA13 $<$ 1073.033
	MM31 $\in$ [0.185, 0.355)		
THEN	MM116 = 0.6 (0.200)	THEN	MM116 = 0.6 (0.134)

Table 6.8: Exemplary rules generated by means of the TDF method with lower values of methane concentration.

The presented example of the pessimistic and optimistic rule quality analysis shows that it can bring additional knowledge about the generated rules and allows to compare the rules more deeply. Moreover, the descriptive abilities of rule-based models allow to conclude about the process directly based on the rules body.

		R3		R4	
		Precision	Coverage	Precision	Coverage
Rule quality	pessimistic	0.893	0.740	0.714	0.195
	standard	0.903	0.758	0.755	0.211
	optimistic	0.913	0.775	0.794	0.228

Table 6.9: Quality of the rules from Table 6.8.

6.1.4. Conclusions

An approach to methane concentration prediction was presented, in which the readings of the sensor being the subject of forecasting are not included into the model. Several prediction methods were analyzed in this task. The analysis was performed on real life data consisting of weekly measurements containing methane concentration collected at the coal mine.

The results showed that the best prediction quality was delivered by the TDF method with a single target value which was introduced in this work. Some interesting characteristics of this method were also presented by means of the error distribution analysis.

The TDF method generates regression rules with a single value in the conclusion (calculated differently than in the works [59, 109]). Such rules deliver easily interpretable knowledge about the analyzed phenomenon, which is an additional advantage of this method. A new approach to pessimistic and optimistic evaluation of such rules was also presented in the work and evaluated on two exemplary rules generated for the given task.

The results of algorithms with the linear model in the conclusions show that the application of a linear function does not always lead to a more accurate prediction. However, the TDF method with the linear model still allows to obtain a lower prediction error than other methods with the same model type.

6.2. Seismic hazard prediction

In the previous chapter, the attention was focused on the research on the methane concentration prediction that may lead to a methane explosion and can jeopardize the safety of the miners working underground. However, this is not the only threat to which the miners are exposed. Another threat is a seismic hazard. In mines the seismic hazard is mainly understood as a high energy destructive tremor which can cause rockburst, which, in turn, can threaten the lives of the miners, and lead to the destruction of the equipment and longwalls. Due to such risks it is crucial to anticipate threats and warn of their possible occurrence early enough. The problem is that the seismic hazard is one of the hardest natural hazards to detect and predict.

The problem of predicting the seismic hazard is undertaken in the literature mainly in the context of classification tasks where the response is usually defined as a "hazardous" or "non-hazardous" state. It is worth mentioning such approaches as: prediction tremors using artificial neural networks [61, 71, 99], prediction data clustering techniques [66], probabilistic analysis [70] that predict the energy of future seismic tremors emitted in a given time horizon, or the linear prediction method [65] that predicts the total value of seismoacoustic and seismic energy.

The aim of this work is to adapt the presented regression rule induction algorithms to predict the total of energy of tremors. The verification of prediction abilities will be carried out in relation to a number of publicly available methods in Weka [127], in the R environment [98], or methods from other sources.

6.2.1. Data set

This study concerns the analysis of the measurements collected by the geophysical station supporting system called Hestia [101]. This main task of this system is to gather and visualize data from seismic activity monitoring systems located in a coal mine. An important objective of the system is to assess possible rockburst hazards for each longwall separately on the basis of seismic and seismoacoustic methods.

The data set used in this study was collected with the use of two time horizons: shift (eight hour) one and one-hour horizon. However, taking into account the schedule of the mining work, the shift horizon is more important. The data set is derived from the Mysłowice-Wesoła coal mine from the longwall Sc503, which is threatened with rockbursts because of its geological structure. The task was to predict the total seismic energy of the recorded tremors and energy recorded by the GMax (geophone which records maximum energy during the process of data aggregation) which will be released in the next hour or in the next eight hours (shift) depending on the time horizon of the data set. The description of attributes is presented in Table 6.10.

Each data set was divided into two disjoint subsets. The first 70% of examples consisted of a training set for the model building, and the last 30% of a data created test set for the model evaluation.

6.2.2. Experiment and experimental settings

The *Top-Down* algorithm and *Top-Down Fixed* modification (the best results from the group of TD, TDF, BU and BUF algorithms were obtained for TD and TDF algorithms) were compared with several methods available in Weka 3.6.11 [127] and in the R 3.1.2 environment [98]. The attention has been paid to the methods that, similarly to rule-based ones, are able

Attribute name	Attribute type	Description	Variable type
tremor energy	numerical	prediction of total seismic energy in next hour/shift (depends on the time horizon of data set)	dependent
seismic	nominal	hazard assessment made by seismic methods	independent
seismoacoustic	nominal	hazard assessment made by seismoacoustic methods	independent
comprehensive	nominal	comprehensive hazard assessment	independent
shift	nominal	information about type of a shift (coal-mining or preparation shift)	independent
genergy	numerical	seismic energy recorded within the previous shift by the most active geophone (GMax) out of geophones monitoring the longwall	independent
gpuls	numerical	number of pulses recorded within the previous shift by GMax	independent
gdenergy	numerical	deviation of energy recorded within the previous shift by GMax from average energy recorded during the eight previous shifts	independent
gdpuls	numerical	deviation of a number of pulses recorded within the previous shift by GMax from average number of pulses recorded during the eight previous shifts	independent
ghazard	nominal	result of shift seismic hazard assessment in the mine working obtained by the seismoacoustic method based on registration coming form GMax only	independent
nbumps	numeric	the number of seismic bumps recorded within the previous shift	independent
nbumps2	numeric	the number of seismic bumps (in energy range [10^2,10^3)) registered within the previous shift	independent
nbumps3	numeric	the number of seismic bumps (in energy range [10^3,10^4)) registered within theprevious shift	independent
energy	numeric	total energy of seismic bumps registered within the previous shift	independent
maxenergy	numeric	the maximum energy of the seismic bumps registered within previous shift	independent

Table 6.10: Description of the attributes in the seismic data set.

to express the data model in a comprehensible form and, because of the assumption of the experiment, allow us to get the same type of conclusion (a single target value or a linear model). The methods that were applied and the information about the type of prediction are listed in Table 6.11. All algorithms were run in their default configuration, except RegENDER for which the number of rules was set to 50 (the algorithm was run with different values of rules, but the results do not differ).

Id	Conclusion type	Description	Tool
CART	single value	Classification and regression trees (rpart) [13]	R
Cubist	linear model	Rule-based predictive models [76]	R
M5P	linear model	M5 trees [95]	Weka
M5RULES	single value	M5 rules [51]	Weka
SeCoReg	single value	Separate and conquer rule induction algorithm [34]	External source
RegEnder	single value	Rules constructed using boosting technique [24]	External source

Table 6.11: Reference methods utilized in the analysis.

6.2.3. Results

The results of the analysis are presented in Table 6.12 for comparison of methods with a single target value in the conclusion and in Table 6.13 for methods with linear models. The performance of each algorithm is described by the root relative-squared error (RRSE).

Prediction task	Algorithm	RRSE	Correlation
Hourly predicting	TD C2 Coverage	87.70	0.568
	TDF C2 Coverage	86.15	0.550
	CART	**79.37**	**0.610**
	SeCoReg M-Estimate	83.34	0.569
	M5Rules	82.68	0.564
	RegENDER-50	82.78	0.572
Shift predicting	TD C2 Coverage	70.53	0.732
	TDF C2 Coverage	57.35	0.818
	CART	56.72	0.824
	SeCoReg M-Estimate	60.05	0.798
	M5Rules	**54.49**	**0.840**
	RegENDER-50	56.42	0.832

Table 6.12: sc503

It can be seen that the error and the correlation coefficient between the predicted and actual values of the sum of energy of tremors and energy of emission seismoacoustic are lower for the shift horizon than for the hourly prediction. The best results for the single target value prediction were achieved by the CART algorithm for both time ranges. However, the result obtained for the shift prediction by the TDF algorithm does not differ too much from the best value. In addition, it is worth noting that the values of the correlation coefficient for the best four algorithms (including TDF) are higher than 0.8 and lower than 0.84. In the case of the hourly predicting task, the t-test refused the null hypothesis, which is that both algorithms perform equally well, between the best result for the TDF C2 Coverage method and the CART method (at the 5% significance level and with the p-value 0.026). However, in the case of shift predicting the null hypothesis cannot be rejected (for the TDF C2 Coverage method and the M5Rules method at the 5% significance level the p-value is 0.114). Therefore, this experiment shows that although the proposed method does not give the best results, it is still competitive with other commonly used methods. Moreover, the proposed algorithms have an additional advantage. The output of these algorithms is the rule set with the single target value in each rule. Thus such a form is simple and easy to interpret.

Prediction task	Algorithm	RRSE	Correlation
Hourly predicting	TD C2 Coverage	**77.73**	**0.629**
	TDF C2 Coverage	78.00	0.627
	Cubist	79.96	0.617
	M5P	82.13	0.571
Shift predicting	TD C2 Coverage	**49.72**	**0.867**
	TDF C2 Coverage	55.07	0.834
	Cubist	52.93	0.849
	M5P	55.45	0.834

Table 6.13: sc503 linear model

The results of experiments with the application of linear models were presented in Table 6.13. The comparison was carried out for the algorithms that allow the use of linear models. It is clearly seen that the use of a more complex model allows to obtain a lower prediction error and a higher value of correlation coefficient than for a simple model. For both prediction tasks the t-test refused the null hypothesis between the best result for the TD C2 Coverage and Cubist method (at the 5% significance level and with the p-value 0.019 for hourly predicting and 0.009 for shift predicting). Moreover, in both prediction tasks the best results were obtained for the TD and TDF algorithms. It is also worth to underline that the use of linear models does not always allow for improvement of the prediction error. The examples can be the algorithms M5P and CART where a single target value used in CART outperforms linear models in M5P.

6.2.4. Conclusions

The work presents an approach to seismic hazard prediction with the use of regression rules. Several prediction methods were analyzed in this task. The analysis was performed on real life data consisting of measurements containing the total seismic energy of tremors and energy recorded by a geophone obtained from the geophysical station supporting system in the coal mine.

The results showed that the lowest prediction error was delivered by the methods with the linear conclusion. Although the results of algorithms with the linear model in the conclusion show that the application of a linear function does not always lead to a more accurate prediction (see M5P), the best results (from all presented results) were obtained exactly for the two algorithms TD and TDF with linear models.

The experiments also showed that the methods with a single value in a conclusion are competitive with other commonly used methods. The prediction error and the value of the correlation coefficient obtained for the proposed methods are similar to other values and, in addition, the outputs of the algorithms are simple and interpretable.

7. Conclusions

Solving regression problems with the use of rule-based models is not a trivial problem. Throughout the thesis, we have explored sequential covering rule induction and rule optimization algorithms for solving this issue. The results showed that suitable modifications not only allow the use of these algorithms but also that the algorithms perform similarly to other different state-of-the-art algorithms.

The main goals of this work related to the regression rule induction are: the investigation and the evaluation of covering rule induction algorithms acting on the basis of two entirely different strategies: Top-down and Bottom-up, the introduction of quasi-covering algorithms with the fixed target value in a conclusion of the rule, the transfer of quality measures from classification to regression, and the research on rule optimization algorithms applied during and after the rule induction. The aim of empirical research was to propose the most efficient combination of the rule induction algorithm, the heuristic and methods used at different stages of the induction for solving regression problems. This objective sought to be solved by the choice of the best solutions for each test stage.

Examples of practical applications illustrate the possibility of transferring acquired results to the real regression problems. These examples also show that solving practical tasks requires to define specific field-oriented modifications of sequential covering rule induction and rule optimization algorithms.

This thesis examined many aspects of the rule induction and optimization algorithms. Following, there is a summary of the most important results and findings from each experiment.

1. An application of modifications in the method of determining positive and negative examples covered by the rule for regression allows to apply well-known heuristics from classification to control the process of regression rule induction. The experiment with quality measures for regression allows to identify two most promising (according to the prediction error, the number of rules and the average coverage) quality measures for regression: C2 and Correlation. These experiments also confirmed that regardless of the nature of the algorithm the quality measures retain their characteristics (e.g. tendency to return more general rules). On the other hand, the rule size depends more on the choice of the rule induction algorithm than the choice of the heuristic.

2. Quasi-covering regression rule induction algorithms, acting on the basis of the fixed target value in the conclusion of the rule, allow to obtain good prediction accuracy, especially in the Top-down strategy. Regardless of the rule induction algorithm we have observed that the use of the fixed target value reduces a number of rules in the rule set. The results obtained from experiments on real-life data may also be interpreted as evidence for the prediction of the target value without significant deviation from the actual value. The error distribution is, in turn, more balanced.
3. Research on the separation of rule refinements and rule selection with the use of different quality measures to control both stages shows that the prediction error changes but these changes are not statistically significant. Conversely, studies revealed a significant impact on the number of induced rules from $+10\%$ and -23% to $+77\%$ and -55%. The experiment proved that the separation of stages may have an impact on the outcome of the regression rule induction algorithm.
4. The main conclusion that we draw from the experiment with pre-pruning methods is that the application of the Hill climbing approach is a better choice mostly because the Tabu hill climbing algorithm leads to the induction of more rules. It is still an open question whether the introduction of the parametrized method would have improved the results.
5. Rule filtering algorithms, after applying appropriate modifications, can be successfully used to reduce a number of redundant regression rules in the rule set without significant increase in the value of the prediction error. The redundancy of the rule was examined in two aspects. We can observe that the group of algorithms that focus on the optimization towards the best prediction accuracy in some cases significantly decrease the prediction error but at the cost of higher value of percentage of test examples that were covered by the default rule. The lowest level of reduction in the number of rules was observed for the Inclusion algorithm while the highest reduction was observed for two algorithms: Disjoint-0.9 and ForwBack. Research on various aspects of filtering algorithms allowed us to identify the most universal Coverage algorithm, which reduces a regression rule set by 16%-40%.
6. The visualization of the filtration process allows to conclude about the behaviour of rule filtering algorithms. We have observed similarity of performance curves for algorithms working with a different definition of criterion of uselessness of rules. It can be noted that the rule filtering carried out with respect to the training set at a certain point does not improve and even leads to deterioration of the rule set quality with respect to the test set. Interesting results were also observed for the combination of filtration methods. The successive filtration with the use of two methods: Forward and Backward (labelled as ForwBack) hardly ever changes the value of the prediction error throughout the filtration process, while it still improves the reduction ratio.

7. Research on the conflict resolution methods has shown that the algorithms based on measures like mean, median or heuristic, and not on voting schemes like in classification, may be applied for the problem of solving conflicts in regression. We noticed that the conflict resolution methods lead to a statistically significant different prediction error of the rule set. We can clearly observe that two methods (mean of coverage and intersection of coverage) perform better than others. In both quasi-covering rule induction algorithms the newly proposed algorithm of intersection of coverage even outperforms other methods. Furthermore, we observed that the max rule quality method always provides the worst results of the rule set. Thus, we can conclude that one rule in the case of unordered regression rule set does not allow to obtain accurate prediction. An interesting observation is that it is likely due to the overlapping of different rules between which the true target value lies. The principle of the intersection of the coverage algorithm seems to prove this thesis.
8. The introduction of the confidence interval for the analysis of regression rules gives a broader view for the rule evaluation. The evaluation of the rules can be viewed in two ways. First, the confidence intervals help to determine the strength of a description ability, e.g. giving the information about the range of the rule coverage. Second, the pessimistic, standard and optimistic values of the quality measure imply the role of the rule in the predictive regression model. The experiment showed that the analysis of the confidence intervals can be used in an additional, possible to be performed by a domain expert, assessment of a single rule, many rules, in particular similar rules, or to evaluate rules in the parent-child scheme when one of the rules extends another.
9. It was shown that the proposed algorithms for regression rule induction are comparable to other state-of-the-art algorithms. Moreover, an application of rule filtering algorithms is justified as it retains a good predictive ability of the rule set as well as significantly reduces the number of rules which improve the readability of the model.
10. The use of linear models (although the subject has not been widely addressed in this thesis) led to a reduction in the prediction error by 10% to 20% with respect to the single target value. This experiment confirmed that the predictive ability of the set of regression rules can be easily improved but at the cost of the readability of the model.
11. Examples of practical applications of regression rule induction algorithms show that the presented algorithms, methods and heuristics can be successfully adapted to solve real-life regression problems. Research on the prediction of methane concentration showed that the single target value may lead to better results than linear models. An interesting observation is that the best results were obtained for the quasi-covering algorithm with filtration, which contributed to the reduction of the model to as much as 82%. On the contrary, the seismic

hazard prediction experiment confirmed the accuracy of linear models. In both cases, the regression rules not only contribute to obtain accurate prediction, but above all the use of rule induction algorithms allows to obtain a clear and readable rule-based model, which can be used by a domain expert to extract additional knowledge.

7.1. Further work

Future research on the sequential covering rule induction and rule optimization algorithms may be carried out in several directions.

Research on the quality measures to control the induction process may be conducted towards the verification of the use of the approach of data-driven adaptive selection of rule quality measure as it was presented for the problem of classification [110]. The same data-driven approach may be also applied for the modification of a number of positive and negative examples with the use of confidence intervals. We believe that both solutions may lead to the induction of more accurate and specific rules at the same time.

The separation of rule selection and rule refinement stages seems to be a promising path to a deeper analysis. Although our first experiment does not prove statistically significant differences in the target value, the change in the rule-based model size was unexpected even for us. The possible research directions in the refinement stage could be the development of the method that not only takes into account the current value of the rule quality but also would try to predict the final contribution of the rule in the prediction accuracy of the rule-based model, e.g. based on the information about the importance of elementary conditions. This information could be also used in combination with the Tabu approach to prevent the removal of conditions relevant to the accuracy of the model and not the rule as such.

A huge field of possibilities is in the research on the area of rules and rule sets optimization. First, we noticed that although the lowest prediction error was obtained for methods that focus on the optimization towards the best prediction accuracy but the methods suffer from the production of a large number of unrecognized examples. It seems to us that this problem can be easily solved by re-induction of rules for all uncovered examples. A completely different but equally promising direction of research is the development of a method for the redefinition of rules. The results obtained for classification problems [104] give reason to believe that this approach can work in the case of regression too.

Finally, we think that it is not worth abandoning the Bottom-up strategy. Although the results are not satisfactory, it is worth to look for the cause. The evaluation of descriptive abilities of generated rules may be the first step to undertake. It seems that the generalization phase is also very important for the final form of the rule. Taking into account only one nearest example, might finish the generalization phase too quickly. Thus, our future efforts will focus on developing an algorithm to generalize the rule using more examples from the neighbourhood and all elementary conditions (based on the information about their importance) at the same time.

List of Figures

List of Tables

Bibliography

[1] Thomas Ågotnes, Jan Komorowski, and Terje Løken. Taming large rule models in rough set approaches. In JanM. Żytkow and Jan Rauch, editors, *Principles of Data Mining and Knowledge Discovery*, volume 1704 of *Lecture Notes in Computer Science*, pages 193–203. Springer Berlin Heidelberg, 1999.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499, 1994.

[3] Timo Aho, Bernard Ženko, Saşo Džeroski, and Tapio Elomaa. Multi-target regression with rule ensembles. *J. Mach. Learn. Res.*, 13(1):2367–2407, August 2012.

[4] Aijun An and Nick Cercone. Rule quality measures for rule induction systems: Description and evaluation. *Computational Intelligence*, 17(3):409–424, 2001.

[5] Tim L. Andersen and Tony R. Martinez. Np-completeness of minimum rule sets. In *Proceedings of the 10th International Symposium on Computer and Information Sciences*, pages 411–418, 1995.

[6] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[7] Jerzy Błaszczyński, Roman Słowiński, and Marcin Szeląg. Sequential covering rule induction algorithm for variable consistency rough set approaches. *Information Sciences*, 181(5):987 – 1002, 2011.

[8] Roberto J. Bayardo, Jr. and Rakesh Agrawal. Mining the most interesting rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 145–154, New York, NY, USA, 1999. ACM.

[9] Michael J. A. Berry and Gordon S. Linoff. *Mastering Data Mining: The Art and Science of Customer Relationship Management*. Wiley, 1999.

[10] M.R. Berthold and D.J. Hand. *Intelligent Data Analysis: An Introduction*. Springer, 2003.

[11] Eric Bloedorn and Ryszard S. Michalski. Data driven constructive induction in AQ17-PRE: A method and experiments. In *Proceedings of the Third International Conference on Tools for AI*, pages 30–37, 1991.

[12] Henrik Boström and Lars Asker. Combining divide-and-conquer and separate-and-conquer for efficient and effective rule induction. In Sašo Džeroski and Peter Flach, editors, *Inductive Logic Programming*, volume 1634 of *Lecture Notes in Computer Science*, pages 33–43. Springer Berlin Heidelberg, 1999.

[13] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[14] Ivan Bruha. *Machine Learning and Statistics, The Interface*, chapter Quality of Decision Rules: Definitions and Classification Schemes for Multiple Rules, pages 107–131. John Wiley and Sons, 1997.

[15] Ivan Bruha and Josef Tkadlec. Rule quality for multiple-rule classifier: Empirical expertise and theoretical methodology. *Intell. Data Anal.*, 7(2):99–124, April 2003.

[16] Izabela Brzezinska, Salvatore Greco, and Roman Slowinski. Mining pareto-optimal rules with respect to support and confirmation or support and anti-support. *Eng. Appl. Artif. Intell.*, 20(5):587–600, August 2007.

[17] Peter Bühlmann. Bagging, boosting and ensemble methods. In *Handbook of Computational Statistics*, pages 985–1022. Springer, 2012.

[18] David Christensen. Measuring confirmation. *Journal of Philosophy*, 96(9):437–461, 1999.

[19] Paweł Cichosz. *Systemy uczące się*. Wydawnictwa Naukowo-Techniczne, Warszawa, 2000.

[20] Peter Clark and Robin Boswell. Rule induction with CN2: Some recent improvements. In *Machine learning—EWSL-91*, pages 151–163. Springer, 1991.

[21] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine learning*, 3(4):261–283, 1989.

[22] William W. Cohen. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.

[23] Pawel Delimata, Mikhail Ju. Moshkov. Ju., Andrzej Skowron, and Zbigniew Suraj. *Inhibitory Rules in Data Analysis: A Rough Set Approach*, volume 163 of *Studies in Computational Intelligence*. Springer Berlin Heidelberg, 2009.

[24] Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. Solving regression by learning an ensemble of decision rules. In *International Conference on Artificial Intelligence and Soft Computing, 2008*, volume 5097 of *Lecture Notes in Artificial Intelligence*, pages 533–544. Springer-Verlag, 2008.

[25] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

[26] Włodzisław Duch. What is computational intelligence and where is it going? In *Challenges for computational intelligence*, pages 1–13. Springer, 2007.

[27] Tom Fawcett. PRIE: a system for generating rulelists to maximize ROC performance. *Data Mining and Knowledge Discovery*, 17(2):207–224, 2008.

[28] Eibe Frank and Ian H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 144–151, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[29] Jerome H Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software (TOMS)*, 3(3):209–226, 1977.

[30] Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, pages 916–954, 2008.

[31] Johannes Fürnkranz. FOSSIL: A robust relational learner. In Francesco Bergadano and Luc De Raedt, editors, *Machine Learning: ECML-94*, volume 784 of *Lecture Notes in Computer Science*, pages 122–137. Springer Berlin Heidelberg, 1994.

[32] Johannes Fürnkranz. Pruning algorithms for rule learning. *Machine Learning*, 27(2):139–172, 1997.

[33] J. Fürnkranz, D. Gamberger, and N. Lavrač. *Foundations of Rule Learning*. Cognitive Technologies. Springer, 2012.

[34] Johannes Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13:3–54, 1999.

[35] Johannes Fürnkranz. Workshop on Advances in Inductive Rule Learning. 15th European Conference on Machine Learning, 2004.

[36] Johannes Fürnkranz and Peter A. Flach. Roc 'n' rule learning – towards a better understanding of covering algorithms. *Mach. Learn.*, 58(1):39–77, January 2005.

[37] Johannes Furnkranz and Gerhard Widmer. Incremental reduced error pruning. In *International Conference on Machine Learning*, pages 70–77, 1994.

[38] Francis Galton. Regression Towards Mediocrity in Hereditary Stature. *The Journal of the Anthropological Institute of Great Britain and Ireland*, 15:246–263, 1886.

[39] Dragan Gamberger and Nada Lavrač. Confirmation rule sets. In DjamelA. Zighed, Jan Komorowski, and Jan Żytkow, editors, *Principles of Data Mining and Knowledge Discovery*, volume 1910 of *Lecture Notes in Computer Science*, pages 34–43. Springer Berlin Heidelberg, 2000.

[40] Alan Genz and Koon-Shlng Kwong. Numerical evaluation of singular multivariate normal distributions. *Journal of Statistical Computation and Simulation*, 68(1):1–21, 2000.

[41] Fred Glover. Tabu search—part i. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[42] Fred Glover. Tabu search—part ii. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[43] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.

[44] Ruth Z. Gold. Tests auxiliary to χ^2 tests in a markov chain. *The Annals of Mathematical Statistics*, 34(1):56–74, 03 1963.

[45] Salvatore Greco, Benedetto Matarazzo, and Roman Slowinski. Multicriteria classification by dominance-based rough set approach. *Handbook of data mining and knowledge discovery. Oxford University Press, New York*, 2002.

[46] Jerzy W Grzymala-Busse. Rule induction. In *Data Mining and Knowledge Discovery Handbook*, pages 277–294. Springer, 2005.

[47] Jerzy W. Grzymala-Busse and Wojciech Ziarko. Data mining based on rough sets. In *Data Mining: Opportunities and Challenges*, pages 142–173. IGI Global, 2003.

[48] JerzyW. Grzymala-Busse. Rule induction, missing attribute values and discretization. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 7797–7804. Springer New York, 2009.

[49] Steve R Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14, 1998.

[50] Gunjan Gupta, Alexander Strehl, and Joydeep Ghosh. Distance based clustering of association rules. In *In Intelligent Engineering Systems Through Artificial Neural Networks (Proceedings of ANNIE 1999*, pages 759–764. ASME Press, 1999.

[51] Geoffrey Holmes, Mark Hall, and Eibe Frank. Generating rule sets from model trees. In *Australian Joint Conference on Artificial Intelligence*, pages 1–12, 1999.

[52] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical statistics*, 15(3):651–674, 2006.

[53] Ronald L. Iman and James M. Davenport. Approximations of the critical region of the friedman statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980.

[54] Hisao Ishibuchi and Takashi Yamamoto. Effects of three-objective genetic rule selection on the generalization ability of fuzzy rule-based systems. In CarlosM. Fonseca, PeterJ. Fleming, Eckart Zitzler, Lothar Thiele, and Kalyanmoy Deb, editors, *Evolutionary Multi-Criterion Optimization*, volume 2632 of *Lecture Notes in Computer Science*, pages 608–622. Springer Berlin Heidelberg, 2003.

[55] Cezary Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 13:189–228, 1993.

[56] Frederik Janssen. *Heuristic Rule Learning*. PhD thesis, Technische Universität Darmstadt, 2012.

[57] Frederik Janssen and Johannes Fürnkranz. Separate-and-conquer regression. In Martin Atzmüller, Dominik Benz, Andreas Hotho, and Gerd Stumme, editors, *Proceedings of LWA2010 - Workshop-Woche: Lernen, Wissen & Adaptivitaet*, Kassel, Germany, 2010.

[58] Frederik Janssen and Johannes Fürnkranz. On the quest for optimal rule learning heuristics. *Mach. Learn.*, 78(3):343–379, March 2010.

[59] Frederik Janssen and Johannes Fürnkranz. Heuristic rule-based regression via dynamic reduction to classification. In Toby Walsh, editor, *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)*, pages 1330–1335, 2011.

[60] J.M. Joyce. *The Foundations of Causal Decision Theory*. Cambridge Studies in Probability, Induction and Decision Theory. Cambridge University Press, 1999.

[61] Józef Kabiesz. Effect of the form of data on the quality of mine tremors hazard forecasting using neural networks. *Geotechnical & Geological Engineering*, 24(5):1131–1147, 2006.

[62] Aram Karalič and Ivan Bratko. First order regression. *Machine Learning*, 26(2-3):147–176, 1997.

[63] Kenneth A. Kaufman and Ryszard S. Michalski. Learning in an inconsistent world: Rule selection in star/aq18, 1999.

[64] G. Keren and C. Lewis. *A Handbook for Data Analysis in the Behaviorial Sciences: Volume 1: Methodological Issues Volume 2: Statistical Issues*. Taylor & Francis, 2014.

[65] J Kornowski. Linear prediction of aggregated seismic and seismoacoustic energy emitted from a mining longwall. *ACTA MONTANA*, 129:5–14, 2003.

[66] Stanisław Kowalik. Prognosis of strong tremors in a mine with the application of fuzzy numbers. In *European Symposium on Intelligent Techniques*, pages Chania, Greece, 3–4 June, 47–49, 1999.

[67] M. Kozielski, A. Skowron, Ł. Wróbel, and M. Sikora. Regression rule learning for methane forecasting in coal mines. *Beyond Databases, Architectures, and Structures, CCIS*, Vol. 521, Springer International Publishing:495–504, 2015.

[68] Stefan Kramer and Gerhard Widmer. Inducing classification and regression trees in first order logic. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 140–159. Springer Berlin Heidelberg, 2001.

[69] Marzena Kryszkiewicz. Fast discovery of representative association rules. In *Rough Sets and Current Trends in Computing*, pages 214–221, 1998.

[70] S Lasocki. Probabilistic analysis of seismic hazard posed by mining induced events. In *Proc. 6th Int. Symp. on Rockburst in Mines "Controlling Seismic Risk". ACG, Perth*, pages 151–156, 2005.

[71] Andrzej Leśniak and Zbigniew Isakow. Space–time clustering of seismic events and hazard assessment in the zabrze-bielszowice coal mine, poland. *International Journal of Rock Mechanics and Mining Sciences*, 46(5):918–928, 2009.

[72] Tony Lindgren. Methods for rule conflict resolution. In *Machine Learning: ECML 2004*, pages 262–273. Springer, 2004.

[73] Tony Lindgren. On handling conflicts between rules with numerical features. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 37–41. ACM, 2006.

[74] Tony Lindgren and Henrik Boström. Resolving rule conflicts with double induction. *Intelligent Data Analysis*, 8(5):457–468, 2004.

[75] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the 4th international conference on Knowledge Discovery and Data mining (KDD'98)*, pages 80–86. AAAI Press, August 1998.

[76] RuleQuest Research Ltd. C5.0. Online documentation http://www.rulequest.com, 2010.

[77] R. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP-69)*, volume A3, pages 125–128, 1969.

[78] R. S. Michalski. Discovering classification rules using variable-valued logic system vl. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, pages 162–172, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

[79] R.S. Michalski. *The AQ15 Inductive Learning System: An Overview and Experiments.* ISG report. Department of Computer Science, University of Illinois at Urbana-Champaign, 1986.

[80] R.S. Michalski. O naturze uczenia się – problemy i kierunki badawcze. In *Informatyka*, number 2. 1988.

[81] R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. *Machine Learning: An Artificial Intelligence Approach*. Springer, Berlin, Heidelberg, 1984.

[82] Ryszard S. Michalski and Kenneth A. Kaufman. The aq19 system for machine learning and pattern discovery: A general description and user's guide, 2001.

[83] Ryszard S Michalski, Igor Mozetic, Jiarong Hong, and Nada Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. *Proc. AAAI 1986*, pages 1–041, 1986.

[84] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[85] Andrew Moore. A tutorial on kd-trees. Extract from PhD Thesis, 1991. Available from http://www.cs.cmu.edu/simawm/papers.html.

[86] Krystyna Napierala and Jerzy Stefanowski. BRACID: a comprehensive approach to learning rules from imbalanced data. *Journal of Intelligent Information Systems*, 39(2):335–373, 2012.

[87] HungSon Nguyen and Dominik Ślęzak. Approximate reducts and association rules. In Ning Zhong, Andrzej Skowron, and Setsuo Ohsuga, editors, *New Directions in Rough Sets, Data Mining, and Granular-Soft Computing*, volume 1711 of *Lecture Notes in Computer Science*, pages 137–145. Springer Berlin Heidelberg, 1999.

[88] A Øhrn, Lucila Ohno-Machado, and Todd Rowland. Building manageable rough set classifiers. In *Proceedings of the AMIA Symposium*, page 543. American Medical Informatics Association, 1998.

[89] Gisele L Pappa and Alex A Freitas. Automatically evolving rule induction algorithms. In *Machine Learning: ECML 2006*, pages 341–352. Springer, 2006.

[90] Zdzislaw Pawlak. *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[91] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 229–248. AAAI Press, 1991.

[92] J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[93] Ross J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81 – 106, 1986.

[94] Ross J. Quinlan. Generating production rules from decision trees. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'87, pages 304–307, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[95] Ross J. Quinlan. Learning with continuous classes. In *5th Australian Joint Conference on Artificial Intelligence*, pages 343–348, Singapore, 1992. World Scientific.

[96] Ross J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[97] Ross J. Quinlan and Ronald L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80(3):227–248, 1989.

[98] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[99] V. Rudajev and R. Číž. Estimation of mining tremor occurrence by using neural networks. *pure and applied geophysics*, 154(1):57–72, 1999.

[100] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, EC '00, pages 158–167, New York, NY, USA, 2000. ACM.

[101] M. Sikora and P. Mazik. A trend towards the better assessment of a seismic hazard - the hestia and hestia mapa systems. *Mechanization and Automation of Mining*, 3:5–12, 2009.

[102] Marek Sikora. Rule quality measures in creation and reduction of data rule models. In *Proceedings of the 5th international conference on Rough Sets and Current Trends in Computing*, RSCTC'06, pages 716–725, Berlin, Heidelberg, 2006. Springer-Verlag.

[103] Marek Sikora. Decision rule-based data models using TRS and NetTRS – methods and algorithms. In James Peters and Andrzej Skowron, editors, *Transactions on Rough Sets XI*, volume 5946 of *Lecture Notes in Computer Science*, pages 130–160. Springer Berlin / Heidelberg, 2010.

[104] Marek Sikora. Redefinition of decision rules based on the importance of elementary conditions evaluation. *Fundam. Inf.*, 123(2):171–197, April 2013.

[105] Marek Sikora and Aleksandra Gruca. Induction and selection of the most interesting gene ontology based multiattribute rules for descriptions of gene groups. *Pattern Recogn. Lett.*, 32(2):258–269, January 2011.

[106] Marek Sikora and Pawel Proksa. Induction of decision and association rules for knowledge discovery in industrial databases. In *International Conference on Data Mining, Alternative Techniques for Data Mining Workshop, Brighton, UK*, 2004.

[107] Marek Sikora and Beata Sikora. Improving prediction models applied in systems monitoring natural hazards and machinery. *International Journal of Applied Mathematics and Computer Science*, 22(2):477–491, 2012.

[108] Marek Sikora and Beata Sikora. Rough natural hazards monitoring. In *Rough Sets: Selected Methods and Applications in Management and Engineering*, pages 163–179. Springer, 2012.

[109] Marek Sikora, Adam Skowron, and Łukasz Wróbel. Rule quality measure-based induction of unordered sets of regression rules. In Allan Ramsay and Gennady Agre, editors, *Artificial Intelligence: Methodology, Systems, and Applications*, volume 7557 of *Lecture Notes in Computer Science*, pages 162–171. Springer Berlin Heidelberg, 2012.

[110] Marek Sikora and Łukasz Wróbel. Data-driven adaptive selection of rule quality measures for improving rule induction and filtration algorithms. *International Journal of General Systems*, 42(6):594–613, 2013.

[111] Andrzej Skowron, Hui Wang, Arkadiusz Wojna, and Jan Bazan. Multimodal classification: Case studies. In JamesF. Peters and Andrzej Skowron, editors, *Transactions on Rough Sets V*, volume 4100 of *Lecture Notes in Computer Science*, pages 224–239. Springer Berlin Heidelberg, 2006.

[112] Dominik Ślęzak. Searching for frequential reducts in decision tables with uncertain objects. In Lech Polkowski and Andrzej Skowron, editors, *Rough Sets and Current Trends in Computing*, volume 1424 of *Lecture Notes in Computer Science*, pages 52–59. Springer Berlin Heidelberg, 1998.

[113] Donald F Specht. A general regression neural network. *Neural Networks, IEEE Transactions on*, 2(6):568–576, 1991.

[114] Katarzyna Stąpor. Using tabu search for feature selection in discriminant analysis. *Studia Informatica*, 35(4):45–58, 2014.

[115] Julius Stecher, Frederik Janssen, and Johannes Fürnkranz. Separating rule refinement and rule selection heuristics in inductive rule learning. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8726 of *Lecture Notes in Computer Science*, pages 114–129. Springer Berlin Heidelberg, 2014.

[116] Jerzy Stefanowski and Daniel Vanderpooten. Induction of decision rules in classification and discovery-oriented perspectives. *Int. J. Intell. Syst.*, 16(1):13–27, 2001.

[117] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(25), 2007.

[118] F. Thabtah, P. Cowling, and Y. Peng. MCAR: multi-class classification based on association rule. In *Computer Systems and Applications, 2005. The 3rd ACS/IEEE International Conference on*, pages 33–, 2005.

[119] Ljupčo Todorovski, Peter Flach, and Nada Lavrač. Predictive performance of weighted relative accuracy. In DjamelA. Zighed, Jan Komorowski, and Jan Żytkow, editors, *Principles of Data Mining and Knowledge Discovery*, volume 1910 of *Lecture Notes in Computer Science*, pages 255–264. Springer Berlin Heidelberg, 2000.

[120] L. Torgo and J. Gama. Regression by classification. In *Advances in Artificial Intelligence*, volume 1159 of *Lecture Notes in Computer Science*, pages 51–60. Springer Berlin Heidelberg, 1996.

[121] Shusaku Tsumoto and Shoji Hirano. Visualization of rule's similarity using multidimensional scaling. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 339–, Washington, DC, USA, 2003. IEEE Computer Society.

[122] S. Venkateswari and R. M Suresh. Association rule mining in e-commerce: A survey. *International Journal of Engineering Science & Technology*, 3, 2011.

[123] Bernard Ženko. *Learning predictive clustering rules : phd thesis*. PhD thesis, Jožef Stefan Institute, Ljubljana, XII 2007.

[124] Bernard Ženko, Sašo Džeroski, and Jan Struyf. Learning predictive clustering rules. In Francesco Bonchi and Jean-François Boulicaut, editors, *Knowledge Discovery in Inductive Databases*, volume 3933 of *Lecture Notes in Computer Science*, pages 234–250. Springer Berlin Heidelberg, 2006.

[125] Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. In *Poster papers of the 9th European Conference on Machine Learning*. Springer, 1997.

[126] Aleksander Wieczorek and Roman Słowiński. Generating a set of association and decision rules with statistically representative support and anti-support. *Information Sciences*, 277:56–70, 2014.

[127] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Amsterdam, 3 edition, 2011.

[128] J. Wojtusiak, R. S. Michalski, K. A. Kaufman, and J. Pietrzykowski. The AQ21 natural induction program for pattern discovery: Initial version and its novel features. In *Proceedings of The 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 13–15, 2006.

[129] Janusz Wojtusiak, Ryszard S Michalski, Kenneth A Kaufman, and Jaroslaw Pietrzykowski. The AQ21 natural induction program for pattern discovery: initial version and its novel features. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 523–526. IEEE, 2006.

[130] Łukasz Wróbel and Marek. Sikora. Censoring weighted separate-and-conquer rule induction from survival data. *Methods of Information in Medicine*, 53(2):137–148, 2014.

[131] Łukasz Wróbel, Marek Sikora, and Adam Skowron. Algorithms for filtration of unordered sets of regression rules. In Chattrakul Sombattheera, NguyenKim Loi, Rajeev Wankar, and Tho Quan, editors, *Multi-disciplinary Trends in Artificial Intelligence*, volume 7694 of *Lecture Notes in Computer Science*, pages 284–295. Springer Berlin Heidelberg, 2012.

[132] Xindong Wu, Vipin Kumar, J. Ross, Quinlan Joydeep, Ghosh Qiang Yang, Hiroshi Motoda, Geoffrey J. Mclachlan, Angus Ng, Bing Liu, Philip S. Yu, Dan Steinberg, X. Wu (b, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, and H. Motoda. Top 10 algorithms in data mining, 2007.

[133] Hui Xiong, Shashi Shekhar, Pang-Ning Tan, and Vipin Kumar. Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 334–343, New York, NY, USA, 2004. ACM.

[134] Y. Yohannes and P. Webb. *Classification and Regression Trees, CART: A User Manual for Identifying Indicators of Vulnerability to Famine and Chronic Food Insecurity*. Microcomputers in policy research. International Food Policy Research Institute, 1999.

[135] Charles Zaiontz. Wilcoxon signed-ranks test. Available online http://www.real-statistics.com/non-parametric-tests/wilcoxon-signed-ranks-test, 09 2014.

[136] Bernard Zenko and Saso Dzeroski. Learning classification rules for multiple target attributes. In *PAKDD*, pages 454–465, 2008.

[137] Wojciech Ziarko and Ning Shan. A method for computing all maximally general rules in attribute-value systems. *Computational Intelligence*, 12:223–234, 1996.