Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science and Engineering

Overview of Doctoral Thesis

Machine Learning-based Feature Ranking and Feature Selection

by

Aleš Pilný

Thesis Supervisor: Miroslav Šnorek
Thesis Co-Supervisor: Pavel Kordík

PhD programme: Electrical Engineering and Information Technology
Specialization: Information Science and Computer Engineering

Prague, February 2015

Thesis Supervisor:
Miroslav Šnorek
Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo nám. 13
121 35 Praha 2
Czech Republic

Thesis Co-Supervisor:
Pavel Kordík
Department of Theoretical Computer Science
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Abstract and contributions

This thesis introduces two novel machine learning methods of feature ranking and feature selection. The methods build on data mining and knowledge discovery techniques. The methods are based on artificial neural network and evolutionary computation. The proposed methods are designed to adapt to the problem,and thus, to provide robust and efficient algorithms and their results in comparison to other used techniques on real world problems. In the past, research in this field has focused on specific kind of problem such as classification or regression. This thesis shows applicability of presented approaches for both types of problems: classification and regression.

Feature ranking and feature selection play an important part of the whole knowledge discovery process. Successful approaches often exploit some expert knowledge specific to a given task, which might be difficult or even impossible to transfer to a different task. The main contribution of the thesis is development of techniques that do not require such an expert input. This is achieved by integrating the construction of the data mining model into the method.

The proposed approach not only delivers the solution, but also derives a mathematical expression that justifies the outcome. This expression is automatically evolved during the data mining process. In case of artificial neural network, the expression represents a data mining model which provides results of classification or regression. In case of genetic programming, the expression represents how the attribute importance was determined and describes a relationship between particular attributes and output variables.

The methods were experimentally evaluated on both, synthetic data, with feature importance known in advance, and on standard real datasets. The quality of the proposed feature selection and ranking is on par with state-of-the-art approaches, in a number of cases even achieving the best performance. Our methods were successfully applied to different real-world disciplines, including anthropology and dental medicine for prediction of age from teeth mineralization, or modelling of oral exostoses.

Abstrakt (Abstract in Czech)

Tato disertační práce představuje dvě nové metody pro určování významností atributů v datech a automatický výběr reprezentativní podmnožiny atributů, čili redukci dat. Prezentované metody využívají principů a technik z oblasti strojového učení, jmenovitě jde o umělé neuronové sítě a genetické programování.

Výhodou metod strojového učení je schopnost adaptovat se na řešený problém. Poskytují robustní a efektivní algoritmy, které konkurují stávajícím technikám na řešených reálných problémech.

V minulosti se výzkum v této oblasti zaměřoval hlavně na specifický typ problému a představované metody se používaly na klasifikační nebo regresní problémy. Tato práce představuje metody pro určování významností faktorů v datech, které jsou schopné pracovat jak s klasifikačními tak regresními problémy. Mnohdy je použití metod pro klasifikaci, regresi či určení významnosti fakotrů natolik specifické, že je nutná jistá expertní znalost ze strany uživatele. To není vždy možné a tato práce se tento problém se snaží vyřešit tím, že využívá a integruje tvorbu modelu pro vytěžování dat a prezentované metody do jednoho algoritmu bez nutnosti specifického nastavování a uživatel tak může použít tyto metody jako takzvané černé skříňky bez nutnosti znalosti jejich obsahu. Výsledkem tak jsou jak klasifikační úspěšnost či chyba regrese, tak ohodnocené vybranné faktory.

Díky využití metod strojového učení je výsledkem použití speciální umělé neuronové sítě také matematický výraz, kterým daná metoda dospěla k výsledku klasifikace či regrese. Při použití genetického programování je výstupem matematický výraz určující významnosti jednotlivých atributů v datech v závislosti na charakteru řešeného problému skrytém v datech.

Prezentované metody byly úspěšně otestovány jak na umělých datech s předem známou významností jednotlivých atributů, tak na reálných datech všeobecně používaných ke srovnávání stávajících a nových metod. Navržené metody byly také úspěšně otestovány na reálných datech z oblasti antropologie a zubního lékařství při řešení problému predikce věku ze stádia mineralizace chrupu a dále modelování výskytu kostních výrůstků v ústech.

Keywords:

Feature Ranking, Feature Selection, Machine Learning, Data Mining, Artificial Neural Network, Genetic Programming, GAME, Classification, Regression

Acknowledgements

First of all, I would like to express my gratitude to my thesis supervisor and co-supervisor. I would like also to thank my readers, for their many constructive comments.

Many other people influenced my work. I would forgot someone therefore I thank You all.

Finally, my greatest thanks to my family who supported me from my date of birth until now. Special appreciation goes to people, who helped me a lot during last several months and also to them, who are with me in their minds. Thank you. It will be such a nice dot after this thesis. I repeat, it will be such a nice dot after this thesis.

Dedication

To my wife Jindřiška, to my parents and to the teacher who had not believed in me.

Contents

List of Figures

Chapter 1

Introduction and Problem Statement

Due to a rapid development of modern technologies, growing amount of data is available every day (internet sharing, community servers, high definition images and video, etc.). Since the development in computer performance is not fast enough to process every piece of information and all the collected or measured data are not needed, the data description and reduction task has become crucial corner stone of a number of data pre-processing and processing tasks.

In machine learning and data mining, the time complexity of many algorithms increases super-linearly with the dimensionality of data. Selecting relevant and removing irrelevant dimensions makes many difficult tasks feasible. An example of applicability of data reduction is in the field of medicine when patients have to undergo many examinations and doctors afterwards make a diagnosis on the basis of all the results. These examinations are often painful, expensive, and stressful. Decreasing the amount of examinations needed for a correct diagnosis reduces the costs and time spent, and is also more comfortable for patients. Next important task is also identification of modifiable measurements. Anthropology is another example how many attributes (e.g. about bones, teeth or skeletons generally) can be collected. But are all of those attributes necessary? What is the right set of attributes for predicting some characteristic? How important are individual data attributes? Is there another set of attributes with similar importance? This thesis is mainly about answers to these questions using new approaches.

Number of measurements, examinations, questions in a poll, etc., all these inputs represent the dimensionality of the processed data set. The task of dimensionality reduction is to reduce the dimensionality of data while preserving relevant information required for each of

the specific tasks. When processing high-dimensional data with various machine learning or data mining algorithms, a phenomenon called "curse of dimensionality" [1] can appear. The problem can arise when increased dimensionality of a problem causes sparse character of data. The phenomenon is therefore a joint problem of an algorithm, which is being applied, and dimensionality of data.

One very useful discipline called feature ranking (FR) [2], which is very close to dimensionality reduction, studies importance of particular attributes of data in relation to a result, which is being searched during a problem exploration. Particular data attributes importance provides very useful information. In case of above mentioned anthropology, one can measure ten proportions of a bone, but thanks to FR they discover that only three of them are the most important in a task of age estimation, other three are less useful and can increase the age estimation accuracy just a little bit and that the rest four proportions do not have an effect on the estimated age. Therefore, FR provides ranking of all attributes in data and can be used as a technique for the dimensionality reduction.

Generally, there are two main approaches in dimensionality reduction [3]. It is feature selection (FS), a method for automated selection of the most relevant attributes, and feature extraction (FE; [4]), a method which is used for extraction of information from the original data set to create subset of new attributes containing all the information from the original data set. FR methods mentioned above can be also used as a dimensionality reduction technique because they are often classified as a special case of FS methods [5], [6]. This relation matches with an idea that only top ranked attributes are selected at the end of FR process [7].

In this thesis, we will focus on the FS and FR methods for the dimensionality reduction, where relevant attributes[1] are selected from the data set (FS) and/or importance (and finally ranks) is assigned to all attributes in the data set (FR).

1.1 Motivation and general considerations

In a field of data mining researchers often have a data set and face a demanding task of its classification. In practice, a data set is of finite sample size. Therefore, the true probability

[1] In the literature an "attribute" is also called as a feature, factor, input or variable. As the meaning is in our case the same (one dimension in a data set), we will interchange these names according to the context.

distribution of data is unknown and its accurate estimation is difficult. If we had all the data samples, we would be able - thanks to the optimal Bayes decision rule [8, 9] - to predict most probable class for a given sample based on the knowledge of the probability distribution describing the data. The Bayes decision rule is a monotone function of the number of features [10, 8], so the probability of the classification error can only increase for the reduced dimensionality. On the other hand, adding more features does not increase the classification error. Hence, the more features the better. This theoretical case is unfortunately far away from practice. As standard learning algorithms do not behave exactly according to Bayesian theory [11], a classifier build on a subset of features can achieve a better performance (higher classification accuracy) than when all features are used for classification.

Dr. R. L. Plackett commented already in 1984 on Dr. Miller's paper regarding Selection of Subsets of Regression Variables [12] saying that "If variable elimination has not been sorted out after two decades of work assisted by high-speed computing, then perhaps the time has come to move on to other problems". Nowadays, more than 30 years later is this statement of much higher importance and could be extended also into a context of classification. In contrast to Dr. Plackett, we believe that there is a way to sort the problem out, at least to a certain level.

The most frequent general reasons for feature ranking and feature selections mentioned in literature (e.g. [6]) are:

- improve model performance and avoid over-fitting
- build faster and more cost-effective models
- gain deeper insight into the underlying processes that generated the data
- loss of information due to an improper data reduction, thus also a lower model performance

In context of this thesis, we add also another important points: data adaptive dimensionality reduction and ranking for achieving of better model performance and getting possibly better problem description

When using distinct learning methods, a different set of features providing best model performance can be obtained. It is also not clear whether the model performance or

a conjunction of the highest feature relevance and the lowest feature redundancy is the correct objective of feature selection or not. Having more than one optimal subset leads naturally to a question which one is the best.

1.2 Goals of the Thesis

The thesis is focused mainly on data adaptive dimensionality reduction. The main goals are as follows:

- To investigate building phase of a data mining model called Group of Adaptive Models Evolution (GAME, [13]) and devise how to assign importance to particular features as well as ranking of features based on the created model.
- To design a classifier model dependent feature scoring function for feature importance and feature ranking determination using Genetic Programming (GP, [14, 15, 16]). In particular, we are interested in a scoring functions dependet on a given data set to provide a data adaptive feature ranking and selection method. The further benefit may be the data specific scoring function itself as well as the classification accuracy obtained during the run of the method. Properties of the designed algorithm have to be examined as well.

Investigation of the area of data adaptive feature selection and feature ranking methods has been chosen as we assume that this approach can provide a closer problem description and thus may help to gain a higher model performance than the methods of the state of the art. Moreover, simplification of a data mining task by minimizing of a role of expert can be a significant advantage of our approach, because there would not be a need to choose an appropriate dimensionality reduction technique and following classifier. It is evident that due to the dependence on a model building phase, this approach could have higher computational requirements in comparison to some of the state of the art approaches. Nevertheless, we believe that this deficiency can be balanced out by the model performance. The higher computational requirement could limit this approach to a certain level of dimensionality. There is also a risk of over-fitting. All these possible issues have to be discussed and avoided by a proper experimental evaluation. It is expected that the key thesis open problems will lead to necessity to combine results of multiple feature

ranking or feature selection runs. Thus, a sub-task of this thesis is to find the best available function combining results of multiple runs of feature selection or feature ranking together.

Last goal of this thesis is to utilize above described methods in current real-world problems in anhropology, described later in detail in section 5.3.2.4 and 5.3.2.5.

1.3 Organization of the thesis

The thesis is organized into three main parts: Theoretical Background, Our Proposed Methods and Experiments and Results. The theoretical background 2 includes introduction to the current state-of-the-art in a field of FR and FS and theory to machine learning.

The second part contains theoretical chapters of my newly proposed approaches based on machine learning (chapter 3 and chapter 4)

The third part, Experiments and Results is divided into two main chapters, where my machine learning based approaches' results are presented (artificial neural network-based approach 6 and genetic programming based approach 7).

Chapter 2

Background and introduction to the State-Of-The-Art

In this chapter a general overview of data mining is presented. The tasks of feature ranking and feature selection are defined as well as an introduction to the state of the art in this field.

2.1 Definitions and terminology

Crucial part of this thesis is term "data". A basic definition of data describes it as a matrix of m vectors, where each vector consists of n elements as it is shown in figure 2.1 Such a data matrix is often referred to as a data set. The i-th element of each vector represents one feature of the data and it is generally called an attribute (variable, factor or input are also in use). The term instance is used for j-th vector of elements. According to our medical example (mentioned above in chapter 1) each patient is an instance and each type of examination (measurement) is a feature. In this work, the terms feature and attribute represent the same entities and are interchangeable. In general, the data set represents a sample of data describing a specific problem. There are different types of techniques from a field of data mining [17] intended for a specific character of data. Each technique analyses a particular problem such as classification, regression, prediction, clustering as described e.g. in [18]. For example, the data analysed by clustering technique does not need to contain the output feature described in figure 2.1, whereas classification

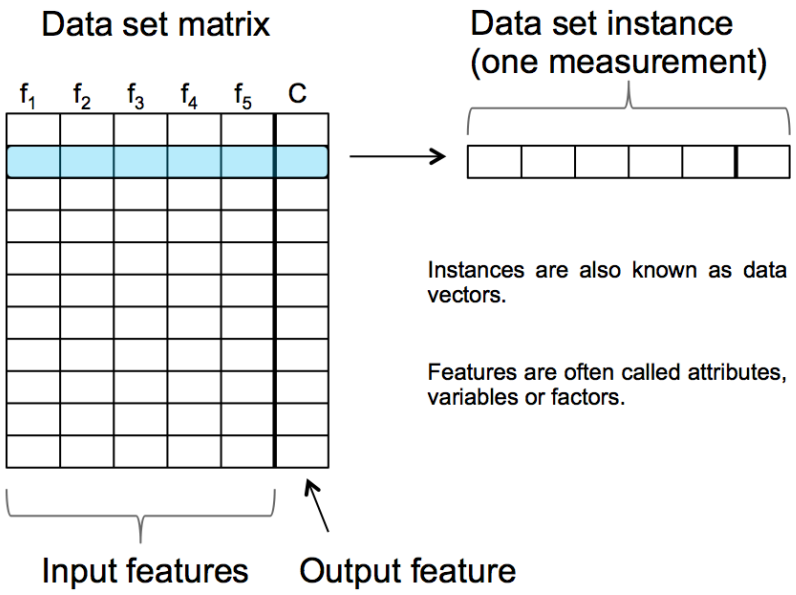

Figure 2.1: Data matrix, vector and feature scheme. The output feature is often called dependent, class or output variable.

and regression techniques do. In the following section we focus on a field of data mining in general.

2.2 Data Mining and Machine Learning

Data mining is an integral part of a complex knowledge discovery process. In figure 2.2 Fayyad et al. in [17] showed the whole process of Knowledge Discovery (KD) from raw data to a knowledge about this data.

This process consists of four main steps starting with the selection of the target data and its preprocessing, where it is usually necessary to adjust the data (e.g. normalizing, replace missing values, etc.) and to reduce data for next processing (select relevant features and/or reduce the number of instances). This step is the first option where one can apply methods of FR and FS. Third stage of KD (according to [17]) consist of matching the goals of the KD process to a particular data-mining method (for example, summarization, classification, regression, clustering, etc.) and exploratory analysis and model and hypothesis selection: choosing the data mining algorithm(s) and selecting method(s) to be used for searching

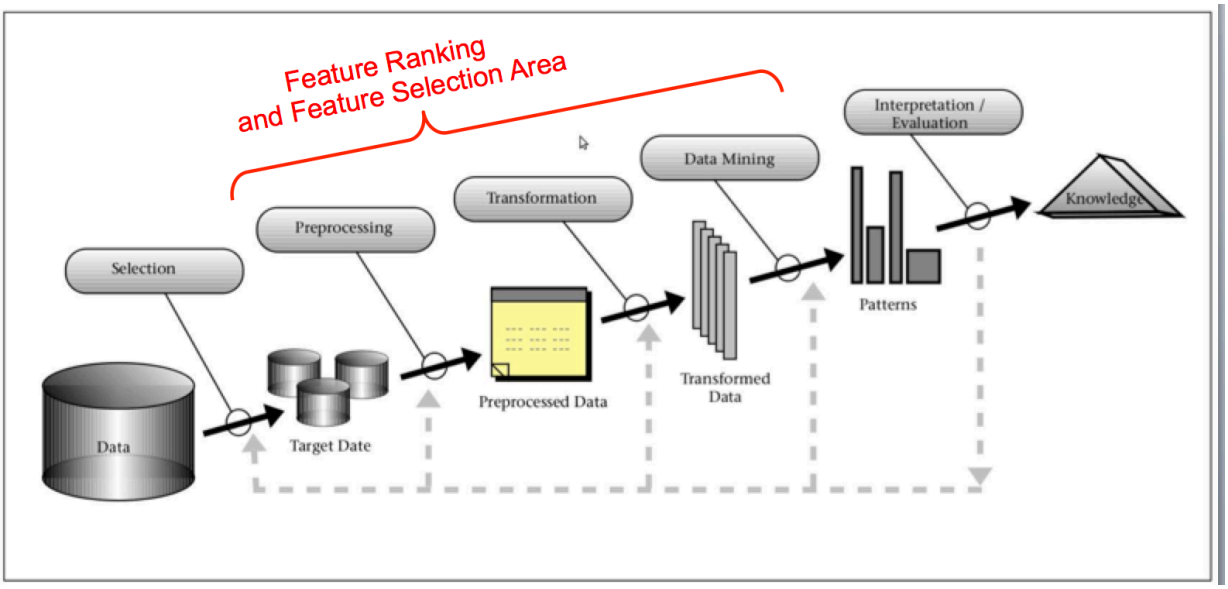

Figure 2.2: The knowledge discovery process incorporating data mining step. Adapted from [17].

for data patterns. Custom application of these methods is known as data mining (in other words searching for patterns of interest) and with results (mined patterns and its interpretation), it is the final step in KD. Second possible application of FR and FS methods is during the data mining process.

Therefore, data mining (sometimes called exploratory data analysis) can be defined as a collection of methods using specific algorithms for extracting patterns from data. Fayyad, Piatetsky-Shapiro and Smyth extend in [17] this description to "the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data".

From our point of view, the "collection of specific algorithms for extracting patterns from data" contains algorithms for machine learning (ML) which are primarily focused on automatic learning to recognize complex patterns and make intelligent decisions based on data [19].

There are three main kinds of learning algorithms in ML: supervised, unsupervised and semi-supervised. In supervised learning, every instance in data set is represented by using the same set of features. The features may be continuous, categorical or binary and instances are given with known labels (the corresponding correct outputs) [19].

While unsupervised learning algorithm works only with unlabeled instances (typical ex-

ample is clustering where similar instances are grouped together and dissimilar ones separated), in semi-supervised learning a small number of instances is labeled and the majority is unlabeled. In this case, both approaches (supervised and unsupervised learning) are combined together to make this kind of learning more efficient. Apart of these types of learning, a data distribution changes in time may influence a task of data mining. Thus, in case that a model is capable of reflecting the changes in evolving dataset, it may produce better performance.

Nowadays, there is a huge number of different machine learning (ML) algorithms, often inspired in nature around us. In this thesis we use two machine learning techniques for feature ranking and feature selection: Artificial Neural Networks and Genetic Programming (sub-part of evolutionary algorithms). These ML techniques are described in detail in next two sections. A general overview of other ML techniques is described in detail e.g. in [18, 20, 21].

2.3 Feature Ranking and Feature Selection

Research in a field of FR and FS methods rapidly expanded over past several decades and many different approaches have been published. Feature selection can be stated as the search for a sufficiently reduced subset of n features out of the total number of all ones N without significantly degrading (or even improving in some cases) the performance of the resulting classifier when using either set of features [22]. On the contrary, methods of feature ranking do not remove any feature from the original set of features. Feature Ranking assesses individual features and assigns them weights according to their degrees of importance [23].

The definition of Feature Selection can be formalized as: Let Υ denote a set of $n \in N$ observations acquired for the examined process. Let $\Phi_k = \{\phi_i \mid i = 1, 2, ..., k; \phi_i \in \Upsilon\}$ represent a non-empty feature subset of cardinality $k \in N$. Consider a criterion function $C : \Phi_k \rightarrow R$ scoring the quality (e.g., model performance) of feature subsets. Let us assume that the higher the criterion value is, the useful information the features carry.

Feature selection can be viewed consequently as a problem of finding such a combination (subset) of features, which maximizes the criterion function C. The cardinality k of a subset is either prespecified (let us denote this problem of finding as k-parametrized) or

is allowed by many FS methods to be optimized in the course of search (to be denoted k-optimizing).

Analyzing this definition we are facing a problem related to a definition of feature itself. According to [24, 25, 23] there are three types of features: relevant features, redundant features and irrelevant features. Relevant features are those, which are relevant for the further processing in any field of data mining and knowledge discovery with respect to the solved problem (e.g. building of classifier etc.). Some features may be redundant, and do not provide any new information for classification, or irrelevant, hence offer no relevant information at all. A detailed description of feature relevance and redundancy is presented in [7, 26]. The occurrence of these features may influence negatively the classifier design and degrade its final performance. On the other hand, the redundant features often carry important information. Suppose two redundant features f_1 and f_2. If we have two subsets of features (Φ_1 of cardinality n and Φ_2 of cardinality $n+1$) where $f_1 \in \Phi_1$, $f_1 \in \Phi_2$ and $f_2 \in \Phi_2$, the criterion function C will score both subsets of features equally. What subset is better in this case? The one with less cardinality? Suppose further a medical example where f_1 is non-modifiable measurement (e.g. age of a patient) and f_2 is modifiable measurement (e.g. body mass index BMI) as stated in [26]. In this case the subset with cardinality $n+1$ containing both features f_1 and f_2 will be more useful than the f_1, because it contains a feature allowing a feasible change. This example represents a fact, that there are different application demands on feature selection methods. In general, different applications may call for different objectives requiring different approaches. The existence of two subsets with the same criterion function score opens another problematic in feature selection called stability and robustness discussed later in this section.

In knowledge discovery FR and FS methods are used either in data preprocessing stage to omit irrelevant or redundant features thereby to speed up the learning phase, or during this learning phase - in dependency on used approach. FR can also select a subset of feature. Here, one selects the top ranked features, where the number of features to select is specified by the user [27] (k-parametrized methods) or analytically determined [28] (k-optimizing methods). In contrast to feature extraction methods, FR and FS techniques do not alter the original feature representation [6]. This is important for our approach.

2.3.1 Objectives of feature selection and feature ranking

According to a deep overview of FS techniques introduced in [26] on a basis of most influential (cited) papers, the easiest objective of feature selection is to determine the "best" subset of features for statistical analysis or building a machine learning model. Sayes at all [6] defined the objectives as manifold, where the most important ones are focused on model accuracy (to avoid over-fitting and improve model performance), selection of a small subset of features (provide faster and more cost-effective models) and focus on feature relevance (to gain a deeper insight into the underlying processes that generated the data). Kohavi and John [7] defined the objective on the basis of the best classification accuracy only. Pudil at all in [29] were defining the objective and goal to select feature subset without significantly degrading performance of the model. The character of features was taken into account in [25] where the focus was on the most relevant features for use in representing the data. Other definitions are focused on selection of a subset of original features [30], minimal classification error or maximal statistical dependency (selection of the most relevant features to the target class) [31]. Model accuracy and feature relevance were placed at the same level by [32], who stated that "discarding irrelevant or redundant features can improve classification performance". However, according to Hocking[33] the feature selection problem is not well defined, because as he stated "it is apparent that there is not a single problem, but rather several problems for which different answers might be appropriate". Unfortunately, Mr. Hocking did not specified a proper definition and one can only adopt one from the current state of the art.

After almost 40 years, the problem of feature selection is not solved and many another problems were identified in this field. For instance problem of stability and robustness of FS, or a very long time overlooked problem of feature over-selection.

2.3.2 Stability and robustness

Kalousis at al.[34] defined the stability of a feature selection algorithm as the robustness of the feature preferences it produces to differences in training sets drawn from the same generating distribution. Stability quantifies how different training sets affect the feature preferences. Kuncheva introduced a stability index I_S based on cardinality of the intersection and correction for chance in [35]. Limitation of this measure is that it is applicable only to feature selection methods that generate feature subsets of fixed cardinality. Suppose

several subsets of features as a result of one FS method (k-optimized), then the stability index I_S is not usable. Lustgartner modified Kuncheva's index of stability and introduced an adjusted stability measure (ASM) for feature subsets of different lengths in [36] as:

$$ASM = \frac{2}{c(c-1)} \sum_{i=1}^{c-1} \sum_{j=i+1}^{c} S_A(s_i, s_j) \tag{2.1}$$

where c is a number of subsets and $S_A(s_i, s_j)$ represents a similarity for a pair of feature subsets, defined as

$$S_A(s_i, s_j) = \frac{r - \frac{k_i k_j}{n}}{min(k_i, k_j) - max(0, k_i + k_j - n)} \tag{2.2}$$

where s_i and s_j are two subsets of features with cardinalities k_i and k_j respectively obtained from a dataset containing n features. The r is the cardinality of the intersection of the two subsets s_i and s_j and the subtracted fraction from r represents the expected value of the cardinality of the two subsets s_i and s_j.

We use this stability measure in comparison of FS results in this thesis. To use only one stability measure can lead to a misleading conclusions. For instance, properties of a final classifier can be deteriorated by selecting the wrong features. In general, more than one stability measure should be used to desribe obtained FS results properly.

Another posibility to make FS more stable is to use ensemble feature selection techniques. In order to improve the robustness of feature selection, a similar idea as in ensemble learning can be used, where multiple classifiers are combined in order to improve performance. In this work, we construct an ensemble of feature selectors by bootstrapping the data, and creating a consensus feature selector that aggregates the results of the single feature selectors by rank summation [37].

In case the feature subsets are further analysed by domain experts (e.g. by interpretation of results from anthropological point of view), the demand on high feature stability could be wrong. The posibility to interpret more than one FS result of a less stable method may be an advantage in comparison to only one FS result of highly stable method, which has no sence from the expert point of view. This polemic would be good to analyse separately in more detail in the future work.

2.3.3 Problem of over-selection

There is a term over-training in machine learning defining a state where a data mining model has learned "too much" from train data that it "fits" also a noise in data and thus it has considerable influence on generalization when evaluated on test data [18].

There is very similar problem in feature selection where features are selected in wrong order during a process of feature selection. Even worse is a situation, when completely wrong features are selected. The resulting subset of features can decrease a final predictor performance. If this problem is not taken into account, a misleading conclusions can be made, especially in comparison among FS methods as described in [38]

The problem of over-selection can be avoided or eliminated (in case of FS wrappers) or simply identified (in case of FS filters or embedded methods) when an independent part of the original data set is used as a test data for final FS result performance evaluation.

2.3.4 Taxonomy of feature selection and feature ranking

According to [6] and [39] the FS process can be divided into three main techniques: filter, wrapper and embedded approach. Whereas filter approach searches independently for an attribute subset of a model selection step, wrapper methods embed the model hypothesis search within the attribute subset search. This division was not sufficient for Mr. Huang and he introduced a detailed taxonomy of FS methods based on their inputs nowdays in [26]. However, we adopted the filter, wrapper and embedded division for this work. Sayes et al. provided a common taxonomy of feature selection methods, showing for each FS technique the most prominent advantages and disadvantages, as well as some examples of the most influential techniques.

Table 2.1 shows filter methods as fast and independent of the classifier. But both filter approaches (univariate and multivariate) have disadvantage that they ignore interaction with the classifier. Whereas filter approach search for feature subset independently of model selection step, wrapper methods embed the model hypothesis search within the feature subset search ([6]). In this case all possible feature subsets are generated and evaluated on specific model (classifier). To search the space of all feature subsets, a search algorithm is then "wrapped" around the classification model ([6]). Filter and Wrapper selectors are schematically presented in figure 2.3.4.

Table 2.1: Overview of feature selection methods for classification. This table contains division according to three kinds of FS approaches together with advantages and disadvantages of these techniques. For each approach are listed the most common algorithms. Source [6].

Model search	Advantages	Disadvantages	examples
Filter	Univariate		
	Fast Scalable Independent of the classifier	Ignores feature dependencies Ignores interaction with the classifier	χ^2 Euclidean distance i-test Information gain
	Multivariate		
	Models feature dependencies Independent of the classifier Better computational complexity than wrapper methods	Models feature dependencies Less scalable than univariate Ignores interaction with the classifier	CFS (1999) MBF (1996) FCBF (2004)
Wrapper	Deterministic		
	Simple Interacts with the classifier Models feature dependencies Less computationally intensive than randomized methods	Risk of over fitting More prone than randomized algorithms to getting stuck in a local optimum (greedy search) Classifier dependent selection	SFS (1978) SBE (1978) Plus q take-away r Beam search (1988)
	Randomized		
	Classifier dependent selection Interacts with the classifier Models feature dependencies	Computationally intensive Classifier dependent selection Higher risk of over-fitting than deterministic algorithms	Simulated annealing Randomized hill climbing Genetic algorithms
Embedded	Deterministic		
	Interacts with the classifier Better computational complexity than wrapper methods Models feature dependencies	Classifier dependent selection	Decision trees Weighted naive Bayes Feature selection using the weight vector of SVM

In general, the filter methods are quick and scalable but do not take into account a building phase of data mining model, so they use only the intrinsic data property and can miss important relations. On the other site, wrapper methods such as Genetic Algorithms in combination with classifiers evaluate selected subset of features and look for the subset with highest classification accuracy [41]. A disadvantage of this approach can be its potential computation intensiveness and that it can have a high risk of over-fitting. From this point of view the third approach - embedded methods - seems to be better. Embedded method incorporates variable selection process within a model learning process; thereby it provides a specific kind of deterministic finding of attribute subset without need to search the whole space of subsets as in the deterministic wrapper method. An example of embedded method can be Artificial Neural Network [42] or Support Vector Machine [27].

The overview of FS and FR methods in table 2.1 was created about 9 years ago so there is a plenty of another approaches, e.g. Fuzzy and Rough Set approaches, Mixture based FS,

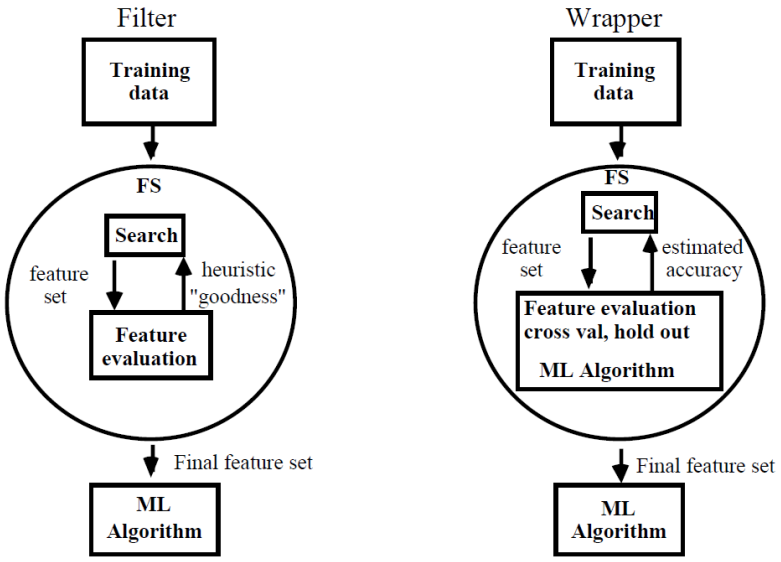

Figure 2.3: Filter and Wrapper feature selectors. Picture published in [40].

Optimal search algorithms, Suboptimal search algorithms, Tabu search based algorithms, Random forests based FS, or several Genetic Programming-based embedded methods. The GP based methods were mainly used to generate new features from the original data set (FE) during an evolution of a classifier. This task was performed without prior knowledge of the probabilistic distribution and exhibitted superior searching power of GP over other techniques on selected data sets [43], [44]. GP was also successfully applied for classifier construction in [45]. In [46] GP evolved decision stumps and ranked features in dependence on how often the particular features were utilized in a process of stumps creation in conjunction with their fitness. Ahmed [47] researched also ability of GP with two already existing embedded FS metrics to select automatically important features and to classify selected subset, so the classification accuracy was a further output of this GP-based algorithm.

Main focus of above described approaches was addressed on specific kind of ML problem such as classification or regression. There do exist also universal data mining and machine learning methods, which are able to solve both of them, e.g. an artificial neural network called GAME [13]. In contrast to [42] where only a subset of relevant features is selected, we propose to use different kind of ANN in order not to get only the FS results, but also the FR result for all selected features in a subset. An advantage of this type of neural network is that it provides an expression of how the output of the network is computed using the selected input attributes.

2.3.5 Example techniques of Feature Ranking and Feature Selection

Generally, there are three different kinds of learning (mentioned in section 2.2) according to which FR and FS methods can be divided. Liu and Motoda described this division in [48] as an application of *supervised*, *unsupervised* and *semisupervised* Feature Selection. As we will see in next sections this division is applicable for Feature Ranking as well.

As we mentioned above, FS methods search for the subset of relevant features. This search has two widely used heuristic sequential strategies, Sequential forward selection (SFS) and sequential backward selection (SBS). The SFS algorithm starts with no feature in a set of already selected features and selects one feature at time until selection of next feature does not improve the subset quality ([48]). Similar the SBS algorithm remove one feature at time until removed feature does not worsen the subset quality according to some studied criterion. Example of methods using this search heuristics are in chapter 2.3.5.

The goal of feature selection is also to avoid selecting too many or too few features than necessary. In practical applications, it is impossible to obtain complete set of relevant features. Therefore, the modeled system is open system, and all important features that are not included in the data set (for what reason ever) are summarized as noise [49]. It is not recommended to select as much features as possible. In fact, even if theoretically more features should provide one with better modeling accuracy, in real cases it has been observed many times that this is not the case. This depends on the limited availability of data in real problems: successful models seem to be in good balance of model complexity and available information. Feature selection tends to produce models that are simpler, clearer, computationally less expensive and, moreover, providing often better prediction accuracy [50].

In statistical analysis, forward and backward step-wise multiple regression (SMR) are widely used [50]. The resulting subset of features generated by adding features until the addition of a new feature no longer results in a significant increment in an R^2 (correlation coefficient) value.

Siedlecki and Sklansky [51] use genetic algorithms for feature selection by encoding the initial set of n features to a chromosome, where 1 and 0 represents presence and absence respectively of features in the final subset. They used classification accuracy, as the fitness function and obtained good neural network results. Mutual information (MI) [2] between

features can be computed by integrating the probability density functions of input and output features. MI is very often used in FS algorithms to distinguish between useful and irrelevant features. Several FS algorithms for the WEKA [18] data mining environment are based on measuring the mutual information of features.

Feature Ranking methods are special case of FS methods where features are ranked according to their relevance. When only a subset of relevant features is needed, an expert or predefined rule has to specify the number of top ranked features for selection. Feature ranking methods can be also divided into three main categories: Filter, Wrapper and Embedded approaches.

The advantages and disadvantages description is identical but this approach has for some methods additional disadvantage - the stop threshold criterion (number of top ranked features). However, as we will see in chapter 3 there is an example of novel feature ranking method which ranks features preselected by embedded feature selection algorithm. On the other hand FR methods are useful for the complex features overview and as well as FS methods do not alter the feature meaning.

In embedded methods results (feature importance and ranks) are mainly obtained as a by-product of learning algorithm - for instance by monitoring which features are being used during a model building phase. There are also FR methods that are directly defined to perform explicit or implicit FS as part of learning. An example are methods based on sub-space mixture modeling (e.g. [52]).

Following sub-sections presents several examples of methods used in data mining. Methods for classification and regression tasks are described as well as methods, which suppose mutual feature independence.

2.3.5.1 Correlation based Feature Selection (CFS)

Marc Hall introduced in [40] new method for feature selection as a correlation based filter approach. This heuristic search for classification problem in supervised learning based on a simple (but important) rule: 'Good feature subsets contain features highly correlated (predictive of) with the class, yet uncorrelated with (not predictive of) each other'. The whole CFS process scheme is in figure 2.3.5.1.

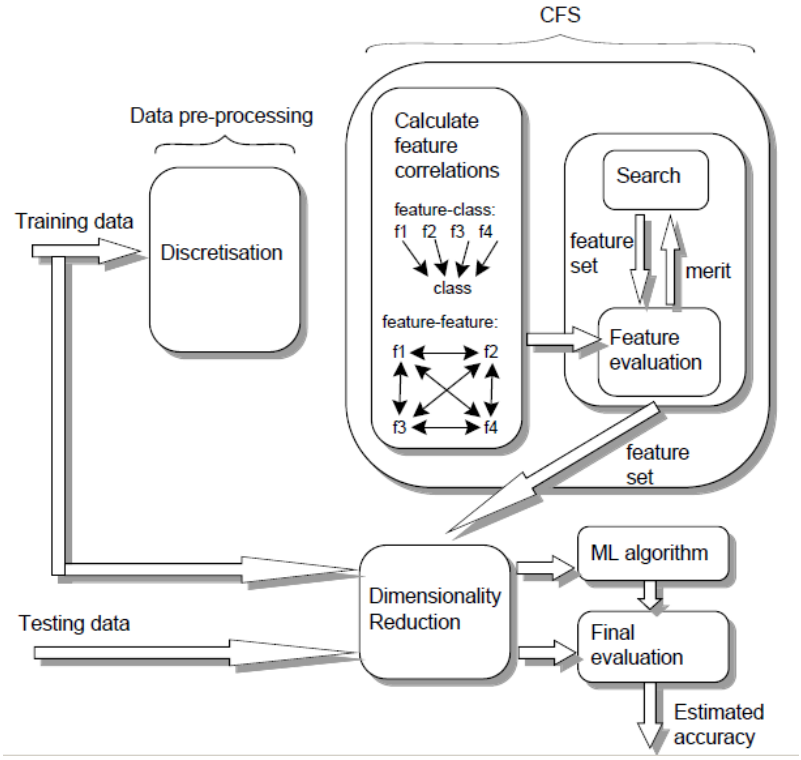

Figure 2.4: The whole process of searching for the best subset of features using CFS algorithm. The process starts with original data set and ends with estimated accuracy of resulting subset.[53]

Hall et al. used for the good feature subset greedy hill climbing search strategy. Main idea of CFS is based on a measure of the feature subset 'goodness'. This measure takes into account the usefulness of individual features for predicting the class label along with the level of inter-correlation among them [53]. The heuristic measure is formalized in equation 2.3:

$$G_s = \frac{k\overline{r_{ci}}}{\sqrt{k + k(k-1)\overline{r_{ii'}}}} \tag{2.3}$$

where k is the number of features in the subset; $\overline{r_{ci}}$ is the mean feature correlation with the class, and $\overline{r_{ii'}}$ is the average feature inter-correlation.

The numerator in the equation 2.3 can be thought of as giving an indication of how predictive of the class a group of features are; the denominator of how much redundancy there is among them [40]. The heuristic goodness measure should filter out irrelevant

features, as they will be poor predictors of the class. Redundant features should be ignored, as they will be highly correlated with one or more of the other features.

2.3.5.2 ReliefF method for Classification (ReliefF)

This method for feature selection is designed for classification problem, but slightly changed can be applied also for regression problem (presented in section 2.3.5.3). This approach assumes the independence of features with respect to the class. In spite of this method is unable to recognize dependencies between features gives good results and is in use.

ReliefF method is extension of basic Relief algorithm which was designed for two class problem and non-missing values and is quite sensitive to noise [48]. All these disadvantages solves algorithm ReliefF shown in pseudo-code below.

ReliefF algorithm for classification

```
1   for i = 1 to N do W[i]=0 end;
2   for q = 1 to m do
3     randomly pick an instance x_k (with class label y_k);
4     for y = 1 to C do
5       find n nearest instances x[j, y] from class y, j=1..n
6       for i = 1 to N do for j = 1 to N do
7         if y == y_k nearest hit?
8           then W[i] = W[i] - diff(i,x_k,x[j, y])/(m*n);
9           else W[i] = W[i] + p_y/(1 − p_{y_k})*diff(i,x_k,x[j, y])/(m*n);
10        end if;
11      end for; j; end for; i
12    end for; y;
13  end for; q;
14  return (W);
```

Input of the ReliefF algorithm is: M learning instances x_k (N features and C classes); probabilities of classes p_y; sampling parameter m; number n of nearest instances from each class. Output for each feature is a quality weight $-1 <= W[i] <= 1$.

In the basic version of Relief algorithm (introduced by Kira and Rendell in 1992) weights

W of features are m-times updated accordingly to special distances between randomly chosen instance and nearest HIT and nearest MISS. Nearest HIT is the nearest instance from the same class and nearest MISS is instance from different class (there are only two classes). Calculation of distance between two instances can be found for instance in [48]. For the multi-class problem ReliefF algorithm searches for n nearest instances (step 5 in ReliefF algorithm above) from each class and the contributions of different classes are weighted with their prior probabilities. This algorithm is also able to work with missing values in instances.

2.3.5.3 ReliefF method for Regression (RReliefF)

Limitation for classification tasks only was removed by Robnink-Sikonja and Kononenko in [54]. The goal of RReliefF (Regressional ReliefF) algorithm in different processing of numerical class variable where nearest MISS and HIT cannot be used in a strict sense as in algorithm ReliefF. RReliefF uses a kind of "'probability"' that two instances belong to two different classes. This probability is modeled with the distance between the values of the class variable of two learning instances. The RReliefF algorithm is listed in pseudo-code below:

RReliefF algorithm for regression problems.

```
1   set all N_dY, N_dF[i], N_dY∧dF[i], W[i] to 0;
2   for q = 1 to m do
3     randomly pick an instance x_k;
4     find indices k_j of n nearest instances, j=1..n;
5     for j = 1 to n do
6       index 0 in diff corresponds to class (regression) variable
7       N_dY = N_dY + diff(i, x_kj, x_k)/n;
8       for i = 1 to N do
9         N_dF[i] = N_dF[i] + diff(i, x_kj, x_k)/n;
10          N_dY∧dF[i] = N_dY∧dF[i] + diff(0, x_kj, x_k) · diff(i, x_kj, x_k)/n;
11      end for; i;
12    end for; j;
13  end for; q;
14  for each feature calculate value of W(F_i)
15  for i = 1 to N do
16    W[i] = N_dY∧dF[i]/N_dY - (N_dF[i]-N_dY∧dF[i])(m-N_dY);
17  end for; i;
18  return (W);
```

Input of the RReliefF algorithm is: M learning instances x_k (described with N features); sampling parameter m; number n of nearest instances. Output for each feature is a quality weight $-1 <= W[i] <= 1$.

$W(F_i)$ value is computed as

$$W(F_i) = \frac{P_{diffcl|diff} P_{diff}}{P_{diffcl}} - \frac{(1 - P_{diffcl|diff}) P_{diff}}{P_{diffcl}} \tag{2.4}$$

where P_{diffcl} denotes the prior probability that two instances belong to different classes and P_{diff} denotes the prior probability that two instances have different feature values. The whole process how is derived equation for $W(F_i)$ can be found in [48]. Algorithm RReliefF has to approximate the probabilities in equation for $W(F_i)$ and calculates the "'frequencies"':

- N_{dY} - sum of "'probabilities"' that two nearest instances belong to different classes;

- $N_{dF}[i]$ - sum of "'probabilities"' that two nearest instances have different feature values

- $N_{dY \wedge dF}[i]$ - sum of "'probabilities"' that two nearest instances belong to different classes and have different feature values.

Finally, from the above "'frequencies"', it calculates the feature qualities $W(F_i)$.

Both algorithms, ReliefF and RReliefF, calculate the quality of features according to equation 2.4, which represents a unified view of the feature quality estimation - in classification and regression.

2.3.5.4 AMIFS method for classification

In 1994 Roberto Battiti introduced method for selection of features by using Mutual Information (MI) in supervised neural net learning ([55]). This approach selects the most relevant k features from an initial set of n features. Method, called MIFS, select the feature that maximizes the information about the class, corrected by subtracting a quantity proportional to the average MI with the previously selected features. When there are many irrelevant and redundant features the performance of MIFS degrades, because it penalizes too much the redundancy (Tesmer and Estevez in [56]). These authors proposed an improved version of MIFS algorithm, denoted as AMIFS that overcome the MIFS limitation in high dimensional feature spaces. An adaptive selection criterion was proposed such that, the trade off between discarding redundancy or irrelevance was adaptively controlled, eliminating the need of an user defined parameter (MIFS algorithm uses a special parameter for controlling the redundancy penalization, whose optimal value is strongly dependent on the problem at hand). Pseudo-code of the AMIFS algorithm is listed below:

AMIFS algorithm for classification

1 (Initialization) set $F \leftarrow$ initial set of n features; $S \leftarrow$ empty set.
2 (Computation of the MI with the output class)
for each feature $f_i \in F$ compute $I(C; f_i)$.
3 (Selection of the first feature)
find the feature f_i that maximizes $I(C; f_i)$; set $F \leftarrow F \setminus \{f_i\}$; set $S \leftarrow \{f_i\}$
4 (Greedy selection) repeat until $|S| = k$:
a) (Computation of the MI between features)
for each $f_i \in F$ and $f_s \in S$ compute $I(f_i; f_s)$.
b) (Selection of the next feature)
choose the feature $f_i \in F$ that maximize
$I(C; f) - \sum_{s \in S} I(f_s; f_i)/N_s \widetilde{H}(f)$;
set $F \leftarrow F \setminus \{f_i\}$ a $S \leftarrow S \cup \{f_i\}$
5 return (S); // the set S containing the selected features

Input of the AMIFS algorithm is: F set of original features; S set of already selected features; the number of features to be selected k;

The main idea of the AMIFS algorithm is to select only such an feature which maximizes expression in step 4 b), where $I(C; f)$ is the Mutual Information between the class variable and the particular feature. Here the right hand term (the sum) is an adaptive redundancy penalization term which corresponds to the average of the MI between each pair of selected features, divided by the minimum entropy between them ([56]).

Disadvantage of this algorithm is still persisting necessity of setting the number of features to be selected. AMIFS works for classification problem only and for speed up of the computational time is necessary to perform binning operation.

2.4 Artificial Neural Networks

Artificial neural network (ANN) can be viewed as a very simplified model of an human brain. According to [57], the human brain is a highly complex, nonlinear, and parallel computer (information-processing system). It has the capability to organize its structural

constituents, known as neurons, so as to perform certain computations (e.g., pattern recognition, perception, and motor control) many times faster than the fastest digital computer in existence today. At human birth, a brain already has considerable structure and the ability to build up its own rules of behavior through what we usually refer to as experience. Indeed, experience is built up over time. This whole process is done by the neural system consisting of neurons and their interconnections. Building up of rules from experience is a process that can be called learning. In parallel to this biological inspiration we can create an artificial model of such neural network to force computers (machines) to learn something from given inputs and known outputs. First of all we have to show what the artificial neural network consists of and how does the process of learning works.

A biological neuron consists of a cell body, dendrites and connections to other neurons, synapses. Such a neuron is an electrically excitable cell that processes and transmits information through electrical and chemical signals. Dendrites propagate the electrochemical stimulation received from other neurons (via synapses) to the cell body.

Similar, an artificial neuron structure corresponds to the biological neuron structure with several simplifications. The structure is displayed in Fig. 2.5 where an artificial neuron consists of m input signal x with synaptic weights w_k, summing junction and activation function φ.

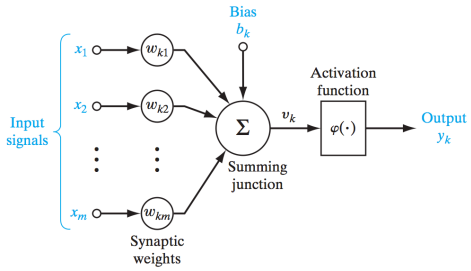

Figure 2.5: An artificial neuron k consists of m input signal x with synaptic weights w_k, summing junction and activation function φ. Output of the neuron is then the output of the activation function. Source [57].

Output of the neuron is then the output of the activation function. Mathematically, we can describe the neuron by equation 2.5 and 2.6:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.5}$$

and

$$y_k = \varphi(u_k + b_k) \tag{2.6}$$

where b_k is an externally applied bias, which has the effect of increasing or lowering the net input of the activation function, depending on whether it is positive or negative, respectively. In [57] the use of bias b_k has the effect of applying an affine transformation to the output u_k of the linear combiner in the model of 2.5 as shown by 2.7.

$$v_k = u_k + b_k \tag{2.7}$$

The activation function, denoted by $\varphi(v_k)$, defines the output of a neuron in terms of the induced local field v. In what follows, [57] identify two basic types of activation functions: threshold function and sigmoid function.

If we have such a defined neuron, then we can build a structure of the neural network. In general, the structure of neural network consists of defined number of layers (usually an input layer, several hidden layers and an output layer). Each layer includes a certain number of neurons. Topology and number of layers usually does not change during time of learning process with several exceptions as we can see for example in section 2.4.1. The network is then ready for a learning phase, which is a base for the second phase, working phase. There are several types of ANNs with different kinds of learning processes.

As we already stated in previous chapter, there are three types of learning. Supervised, un-supervised and semi-supervised. In terms of ANN, the un-supervised learning is a subpart of learning called "learning without a teacher". As the name says, there is no teacher to oversee the learning process, simply because there are no labeled examples in data that could be used for that. The un-supervised learning is being used for finding of hidden structures or patterns in data set. Well known example of ANN which use un-supervised learning is Self Organizing Map (SOM) described e.g. in [58]. Second type of learning without teacher is "reinforcement learning". In reinforcement learning the learning of an inputoutput mapping is performed through continued interaction with the environment in order to minimize a scalar index of performance [57].

On the other hand, in the supervised learning (also called "learning with teacher") the data is labeled and neural network knows what output should be produced on a given input. Therefore error or classification accuracy can be simply calculated after each iteration in

the learning phase. Example of a network using supervised learning, proposed later as an ANN with possibility of FR and FS, is the GAME network (Group of Adaptive Methods Evolution) described in detail in section 2.4.1.

During a learning phase of supervised learning, the weights between neurons are tuned in iterations to learn the network and produce a desired output. As a performance measure of the network, we may think in terms of the mean-square error, or the sum of squared errors over the training sample, defined as a function of the free parameters (i.e., synaptic weights) of the network result. This function may be visualized as a multidimensional error-performance surface, or simply error surface, with the free parameters as coordinates. The true error surface is averaged over all possible inputoutput examples. Any given operation of the system under the teachers supervision is represented as a point on the error surface. For the system to improve performance over time and therefore learn from the teacher (supervisor), the operating point has to move down successively toward a minimum point of the error surface; the minimum point may be a local minimum or a global minimum. A supervised learning system is able to do this with the useful information it has about the gradient of the error surface corresponding to the current behavior of the system. The gradient of the error surface at any point is a vector that points in the direction of steepest descent. In fact, in the case of supervised learning from examples, the system may use an instantaneous estimate of the gradient vector, with the example indices presumed to be those of time. The use of such an estimate results in a motion of the operating point on the error surface that is typically in the form of a random walk. Nevertheless, given an algorithm designed to minimize the cost function, an adequate set of inputoutput examples, and enough time in which to do the training, a supervised learning system is usually able to approximate an unknown inputoutput mapping reasonably well [57]. In a working phase, the network does what it learned for (e.g. classification, pattern recognition etc.).

A possible disadvantage of several ANNs is, that the number of layers and number of neurons in layers has to be manually defined at the beginning (e.g. after some experimental work or by using predefined formulas to calculate appropriate numbers). This is not the case of the GAME network described in th next subsection.

2.4.1 GAME Artificial Neural Network

The GAME network is one of the corner stones for our approach for feature ranking and feature selection in this thesis, thus it is introduced in detail. There are several algorithms for inductive models construction commonly known as Group Method of Data Handling (GMDH) introduced by Ivachknenko in 1966 [59]. The GAME network is inspired in one GMDH based algorithm as shown in next section.

2.4.1.1 Group Method of Data Handling

One of the algorithm for inductive models construction is the Multi-layered Iterative Algorithm (MIA), which uses a data set to construct a model of a complex system. Layers of units transfer input variables to the output of the network. The coefficients of units transfer functions are estimated using the data set describing the modeled system. Networks are constructed layer by layer during the supervised learning stage by induction.

The capability of induction is fundamental for human thinking. It is the next human ability that can be utilized in soft-computing, besides that of learning and generalization. The induction means gathering small pieces of information, combining it, using already collected information in the higher abstraction level to get complex overview of the studied object or process. Inductive modeling methods utilize the process of induction to construct models of studied systems. The construction process is highly efficient, it starts from the minimal form and the model grows according to system complexity. It also works well for systems with many inputs. Where the traditional modeling methods fail, due to the "curse of dimensionality" phenomenon, the inductive methods are capable to build reliable models. The problem is decomposed into small subtask . At first, the information from most important inputs is analyzed in the subspace of low dimensionality, later the abstracted information is combined to get a global knowledge of the system variables relationship.

The original MIA algorithm works as follows. First initial population of units with given polynomial transfer function is generated. Units have two inputs and therefore all pair-wise combinations of input variables are employed. Then coefficients of unit's transfer functions are estimated using stepwise regression or any other optimization method. Units are sorted by their error of output variable modeling. Few of the best performing units are selected and function as inputs for next layer. Next layers are generated identically until the error of modeling decreases to a sufficient level [60]. Which units are performing

best and therefore should survive in a layer is decided using an external criterion [61] of regularity (CR). The CR used in the implementation of the GAME is following:

$$AR(s) = \frac{1}{N_B} \sum_{i=1}^{N_B} (y_i(A) - y_i(B))^2 \rightarrow min \tag{2.8}$$

where the $y_i(A)$ is the output of a GMDH model trained on the A data set (e.g. 2/3 of the original dataset) and the B is the testing data set (e.g. 1/3 of the original dataset), the N_B is the size of the B dataset and the $y_i(B)$ is the output of the GMDH on the testing data set B. The proper regularization is of crucial importance in the GMDH theory. More information on the GMDH topic can be found in the book of A. Muller and F. Lemke [62]

The Group of Adaptive Models Evolution (GAME) algorithm proceeds from the MIA algorithm with several modifications. These modifications of the original MIA algorithm are following: maximal number of unit inputs equals to the number of layer the unit belongs to, inter-layer connections are allowed, transfer function and learning algorithm of units can be of several types, niching genetic algorithm is used to select surviving units and an ensemble of models is generated [60].

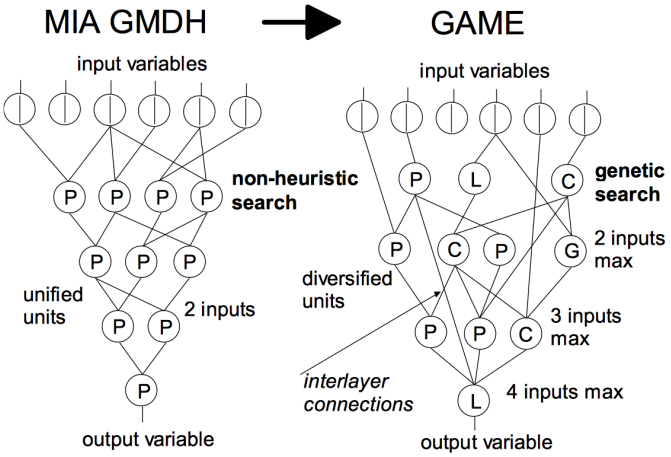

Figure 2.6: Illustration of the basic principle of the GAME artificial neural network. It is based on the MIA-GMDH network. Source [63].

The GAME network is feed-forward artificial neural network and grows as large as needed to solve a problem with sufficient accuracy. The network grows by itself thanks to the Genetic Algorithm. Similar to MIA GMDH described above, it starts from scratch and GA iteratively adds layer by layer as needed. A layer consists of units (neurons) that have

been the most effective in genetic algorithm-based modeling of inter-relationships within the data set, as shown in Fig. 2.6 on the right side. Details of the genetic algorithm used to build the GAME network are described in section 3 separately, because it plays a crucial role for the introduced method for feature ranking.

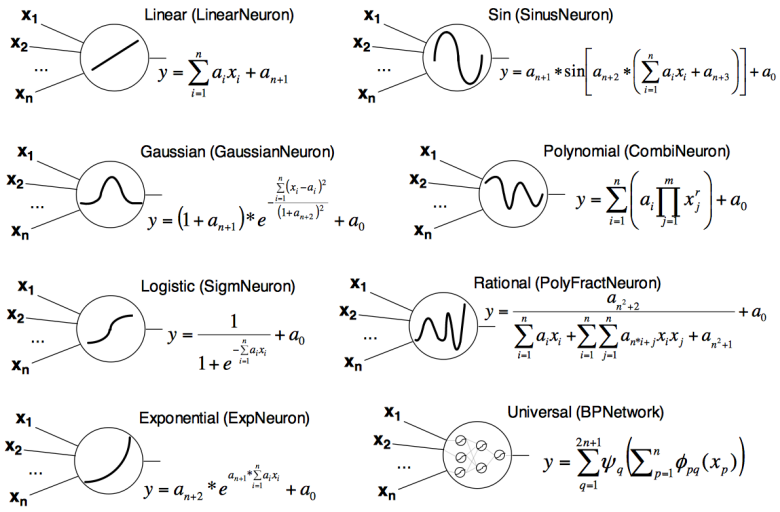

Figure 2.7: Different types of neurons - building blocks of the GAME artificial neural network. Source [63].

The GAME algorithm works with both continuous and discrete variables; its topology adapts to the nature of the data set supplied and it is highly resistant to irrelevant and redundant variables. The more detailed description can be found in [60].

2.5 Evolutionary Computing

Evolutionary computing (EC) is a research area from computer science inspired from the process of natural evolution. Generally, there is a given environment which is filled with a population of individuals that strive for survival and reproduction. The fitness of these individuals relates to how well they succeed in achieving their goal. In EC, the goal of individuals is to solve a given problem in stochastic way of trial-and-error [64].

Evolutionary computing represents a general term for several historical independent streams of a similar approach, namely Fogel-Owens-Walsh's evolutionary programming (EP, [65]), Holland's genetic algorithm (GA, [66],[67]) and Rechenberg and Schefel's evolution strategies (ES, [68]). Since early 1990s they all are known as EC techniques (with different dialects). In more detail is this division described in [64].

Later on, one more technique was introduced in 1985, genetic programming (GP). The first paper to cite is Cramer's [14], but also around 1985 J. Schmidhuber was not aware of this article and reinvented GP. Unfortunately for him, he did not published his work on GP until 1987 in [15].

The whole discipline dealing with population-based techniques is called evolutionary computing. The algorithms involved are termed evolutionary algorithms (EA) with following subareas: EP, ES, GA and GP. In the next subsection 2.5.1, a general evolutionary algorithm scheme and description is presented to show basic principles of EC. In this thesis, the genetic programming plays a crucial role as a technique used for newly proposed feature ranking and feature selection algorithm and is therefore described in detail within this chapter 2.5.2

2.5.1 General Evolutionary Algorithm

The basic idea of any evolutionary algorithm is to maintain a population of individuals that strive for survival and reproduction. As we already introduced in previous section, the fitness of these individuals relates to how well they succeed in achieving their goal. In detail, the general evolutionary algorithm is shown in Fig. 2.8. In this figure, yellow boxes (population, genitors, and offspring) represents sets of individuals. At the beginning, usually a randomly generated initial population of individuals is created. Each individual is evaluated (its fitness) and stop criteria is tested whether the optimal solution was foun

or not. An individual is characterized by its genotype and phenotype. A genotype can be viewed as a structure of information describing that individual. Name of this structure seems to be not unified in literature, moreover it is different also among all types of ES. In GA, the genotype is named chromosome and has a form of an array. This array consists of genes where each gene contains a piece of information describing the individual encoded usually into zeroes and ones (the binary encoding). The phenotype represents the values stored in genotype and can evaluated as a fitness of that individual.

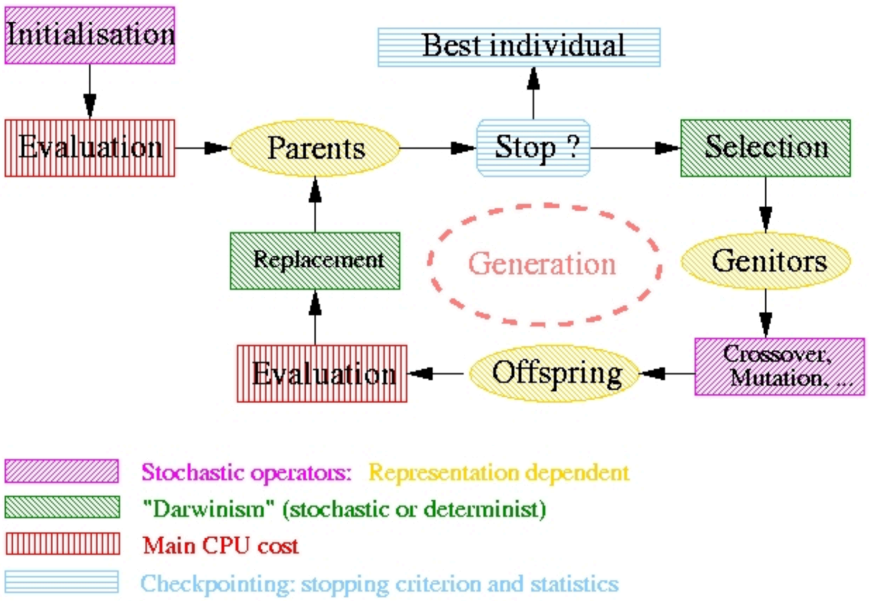

Figure 2.8: Gereral evolutionary algorithm scheme. Source [69].

If any individual did not find the solution, the best solution is stored and the evolution continues (begins). Based on the fitness value of particular individuals, a population of genitors is selected. From this population, the set of offspring is created using recombination (crossover) and mutation operations. Replacement strategy then replaces the original population with the new offspring. For each evolutionary algorithm, the replacement strategy, the crossover and mutation operators, the selection mechanism and the size of genitors, are defined in different way. The crossover is in general an operation, where two individuals interchange a subpart of their genotype (one or more genes in GA) and produce therefore two new individuals. The mutation operation takes one individual and produces a new one with a changed randomly selected part (e.g. a selected gene containing 0 will be changed to 1 in GA).

For more details regarding technique-specific information on genetic algorithm, genetic strategies and evolutionary programming refer to a literature, e.g. [64, 70, 71, 72, 73]. In the next section 2.5.2, the last evolutionary computation technique, genetic programming, is introduced in detail as a background for our proposed GP-based feature ranking and feature selection method described in chapter 4.

2.5.2 Genetic Programming

Genetic programming is the youngest member of the evolutionary algorithm group. Other EAs are being used to solve optimization problems. On the contrary, GP belongs to the machine learning group mentioned in the beginning of this chapter.

In terms of the problem types, most other EAs are for finding some input realizing maximum payoff (Fig. 2.9). The model is known, together with the desired output, (or a description of the desired output) and the task is to find the input(s) leading to this output.

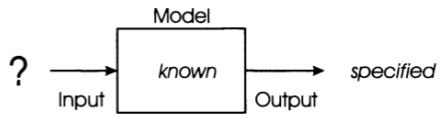

Figure 2.9: Optimization problems. The model is known, together with the desired output, (or a description of the desired output) and the task is to find the input(s) leading to this output. Source [64].

Whereas GP is used to seek models with maximum fit (Fig. 2.9). Here, system inputs and outputs are known and a model of the system is sought that delivers the correct output for each known input. [64]. Clearly, once maximization is introduced, modeling problems can be seen as special cases of optimization. This, in fact, is the basis of using evolution for such tasks: models are treated as individuals, and their fitness is the model quality to be maximized.

The main idea of GP is to evolve computer programs. In figure 2.11, the basic genetic programming scheme, where survival of the fittest is used to find solution, is presented.

According to [70], the GP is among other evolutionary algorithms a proven successful form of weak problem solving in artificial intelligence [23]. At the most abstract level, the

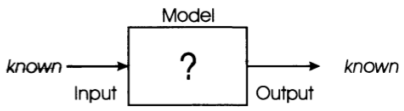

Figure 2.10: Modeling or system identification problems. System inputs and outputs are known and a model of the system is sought that delivers the correct output for each known input. Source [64].

GP is a systematic, domain-independent method for getting computers to solve problems automatically starting from a high-level statement of what needs to be done. In detail, the GP is designed to evolve a population of computer programs.

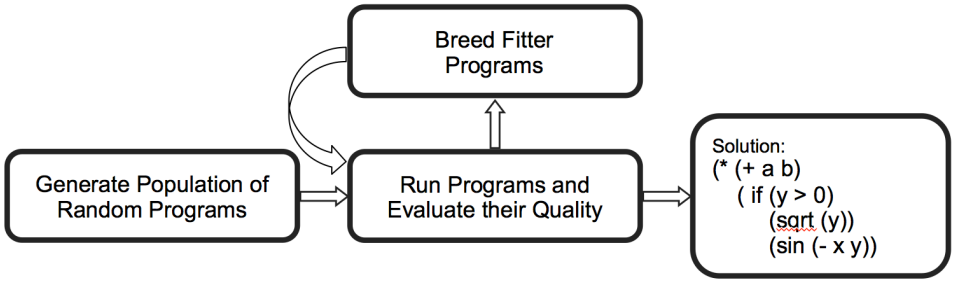

Figure 2.11: The basic genetic programming scheme, where survival of the fittest is used to find solution.

That is, generation-by-generation, the GP stochastically transforms populations of programs into new, hopefully better, populations of programs. Rather than relying on a significant amount of task specific knowledge evolutionary methods only require a credit assignment mechanism that is representation specific rather than task specific. Poli also pointed out, that a representation specific method of credit assignment avoids the need for explicit task knowledge.

There is very important basic idea describing the structure of the program to be evolved as a tree. Thus, there are no loops inside and all possible program trees do indeed describe syntactically correct programs. As an example, there is a parse tree in Figure 2.12. In accordance with the typical GP approach, parse trees are used as genotypes representing formulas. In evolutionary terms, the formula represented by the tree from Figure 2.12 defines the phenotype. If the formula is being used to classify customers, then the classification accuracy represents the fitness of the formula.

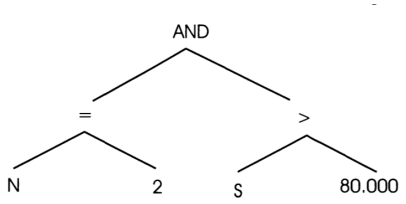

Figure 2.12: Parse tree. Example of a parse tree. Parse trees are used as genotypes representing formulas. Source [64]

.

The tree representation in GP differs from those used in GAs in two important aspects. First, the GP chromosomes are non-linear structures (parse trees), while in GAs and ES they are typically linear vectors. Second, the GP chromosomes can differ in size, measured by the number of nodes of the tree, while in GAs and ES their size, that is, the chromosome length n, is usually fixed.

Parse trees capture expressions in a given formal syntax. Depending on the problem at hand, and the users' perceptions on what the solutions must look like, this can be the syntax of arithmetic expressions, formulas in first-order predicate logic, or code written in a programming language [64], i.e.:

- An arithmetical formula:

$$\pi + r^2 - \frac{1}{2 * R - 1} \tag{2.9}$$

- A logical formula:

$$((a \vee b) \wedge c) \vee d \tag{2.10}$$

- A computer program:

$$\text{j = 1; while(j != 100) j = j + 1;} \tag{2.11}$$

2.5.2.1 Initializing the Population

In GP, initial population is also usually randomly generated. There are two basic methods of random generation of individuals introduced in the beginning of GP: the full and grow methods [70]. In both the full and grow methods, the initial individuals are generated so that they do not exceed a user specified maximum depth. The depth of a node is the

number of edges that need to be traversed to reach the node starting from the trees root node (which is assumed to be at depth 0). The depth of a tree is the depth of its deepest leaf.

procedure: gen_rnd_expr(func_set,term_set,max_d,method)

1: **if** max_d = 0 **or** $\left(\text{method} = \text{grow}\ \textbf{and}\ \text{rand}() < \frac{|\text{term\_set}|}{|\text{term\_set}|+|\text{func\_set}|}\right)$ **then**
2: expr = choose_random_element(term_set)
3: **else**
4: func = choose_random_element(func_set)
5: **for** i = 1 **to** arity(func) **do**
6: arg_i = gen_rnd_expr(func_set, term_set, max_d - 1, method);
7: **end for**
8: expr = (func, arg_1, arg_2, ...);
9: **end if**
10: **return** expr

Notes: **func_set** is a function set, **term_set** is a terminal set, **max_d** is the maximum allowed depth for expressions, **method** is either `full` or `grow`, **expr** is the generated expression in prefix notation and `rand()` is a function that returns random numbers uniformly distributed between 0 and 1.

Figure 2.13: Pseudo code for recursive program generation with the full and grow methods. Source [70]

.

In the full method (so named because it generates full trees, i.e. all leaves are at the same depth) nodes are taken at random from the function set until the maximum tree depth is reached.

Because neither the grow or full method provide a very wide array of sizes or shapes on their own, Koza [16] proposed a combination called ramped half-and-half. Half the initial population is constructed using full and half is constructed using grow. This is done using a range of depth limits (hence the term ramped) to help ensure that we generate trees having a variety of sizes and shapes.

2.5.2.2 Selection

In GP (as well as in most EA), the genetic operators are applied to individuals that are probabilistically selected based on fitness. That is, better individuals are more likely to have more child programs than inferior individuals. The most commonly employed method for selecting individuals in GP is tournament selection described e.g. in [70, 16]. In tournament selection a number of individuals are chosen at random from the population. These are

compared with each other and the best of them is chosen to be the parent. When doing crossover, two parents are needed and, so, two selection tournaments are made.

Thanks to the fact, that tournament selection only looks at which program is better than another (it does not need to know how much better), this effectively automatically rescales fitness, so that the selection pressure on the population remains constant. A system with a strong selection pressure very highly favours the more fit individuals, while a system with a weak selection pressure is not so discriminating. Thus, a single extraordinarily good program cannot immediately swamp the next generation with its children; if it did, this would lead to a rapid loss of diversity with potentially disastrous consequences for a run. Conversely, tournament selection amplifies small differences in fitness to prefer the better program even if it is only marginally superior to the other individuals in a tournament.

2.5.2.3 Crossover and Mutation

There is quiet big difference in GP crossover and mutation from other evolutionary algorithms. The difference is caused by the tree structure of individuals in the population. From a technical view, crossover in GP is a binary operator creating two child trees from two parent trees. Crossover in GP creates offspring by swapping genetic material among the selected parents. The most commonly used form of crossover is sub-tree crossover.

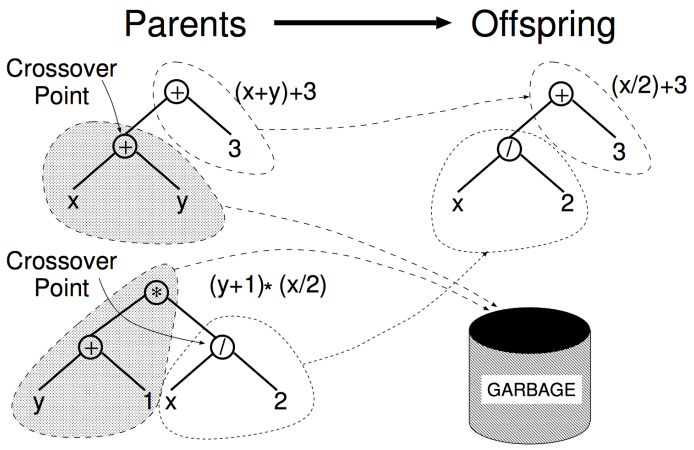

Figure 2.14: Example of sub-tree crossover. Note that the trees on the left are actually copies of the parents. So, their genetic material can freely be used without altering the original individuals. Source [70]

.

Poli [70] describes this crossover as follows; Given two parents, sub-tree crossover randomly (and independently) selects a crossover point (a node) in each parent tree. Then, it creates the offspring by replacing the sub-tree rooted at the crossover point in a copy of the first parent with a copy of the sub-tree rooted at the crossover point in the second parent. This process is shown in Fig. 2.14.

In GP, crossover is parameterized by two probabilities:

- The probability of choosing crossover at the junction with mutation
- The probability of choosing an internal point within each parent as crossover point

In contradistinction to crossover, mutation is the same in GP as in other EAs. The general idea is to create a new individual from an old one through some small random variation. The GP particularity is in replacing the sub-tree by the newly generated started at a randomly selected node by a randomly generated tree as shown in figure 2.15. The process of generation of a new sub-tree is the same with the process of initial population generation.

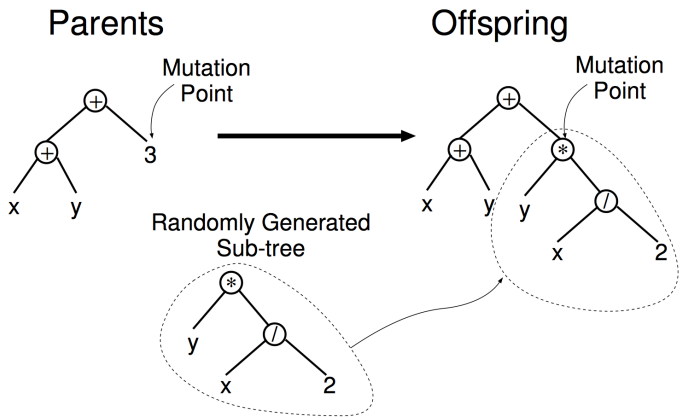

Figure 2.15: Example of GP sub-tree mutation. Source [70]

.

There are similar parameters of mutation as in recombination stated above:

- The probability of choosing mutation at the junction with recombination
- The probability of choosing an internal point within each parent as the root of the subtree to be replaced

Koza [16] advised to set the mutation rate at 0, that means not to perform a mutation. According to [64] and GP community, this settings of a mutation rate is so low (recently, the rate was advised (e.g. in [74]to be 5%), because crossover has a large shuffling effect. Therefore mutation step can be leaved out.

2.5.2.4 Preparation to run GP

There are five basic points (with plenty of possibilities) in a design of the GP algorithm to solve a particular problem. These points have to be defined before running GP. In accordance with [70], we summarized the point as follows:

- terminal set
- function set
- fitness measure
- parameters controlling the run of GP
- termination criterion, and what will be designated the result of the run?

The definitions of several above stated points are very specific and problem dependent (e.g. fitness measure, termination criterion, terminal set, result of the run). On the other hand, the function set can be shared in many different problems solving with some modifications or problem specifics. We define all the above stated points for the problem of feature scoring function in chapter 4.

Chapter 3

Feature Ranking Derived from Data Mining Process

In a field of data mining, there are several techniques to build a data mining model. One group of these techniques are artificial neural networks. To simplify a process of data maining for wider range of researchers we have chosen a promising artificial neural network in order not to cover only a raw data mininig model, but also a field of feature ranking to provide both results, i.e. the data mining model and feature ranking result at the same time.

In this section, a new method for Feature Ranking utilizing information from Niching Genetic Algorithm (FeRaNGA) is proposed. Results of FeRaNGA are derived from information gained during the data mining process. In this case, the data mining process is done by an artificial neural network called GAME which was briefly described in the section 2.4.1.

The niching genetic algorithm (NGA) used in GAME is a cornerstone of the FeRaNGA feature ranking algorithm and needs to be described in detail.

3.1 Niching genetic algorithm

Niching methods [75] extend genetic algorithms to domains that require the location of multiple solutions. They promote the formation and maintenance of stable sub-populations in genetic algorithms. One of these methods is deterministic crowding [76, 77]. The basic

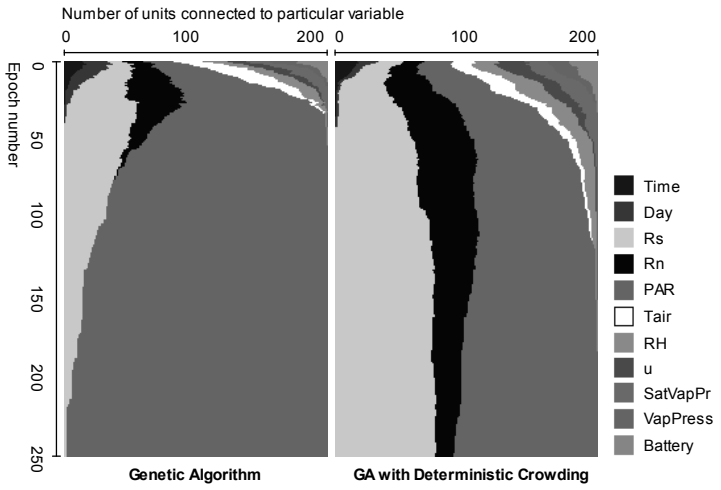

Figure 3.1: GA versus niching GA with DC: first layer of the GAME network (units with single input)

idea of deterministic crowding is that offspring is often most similar to parents. We replace the parent who is most similar to the offspring with higher fitness.

The reason why the deterministic crowding was employed instead of using just simple GA is the ability to maintain multiple sub-populations (niches) in the population. When the model is being constructed, units connected to the most important input would soon dominate in the population of the first layer if a traditional GA (see Fig.3.1) was used. All other units connected to least important inputs would show worse performance on the validation set and disappear from the population with exponential speed.

In inductive modelling, one needs also to extract and use information from least important features and therefore we prefer maintaining various niches in the population. The distance of genes is based on the phenotypic difference of units (to which inputs are connected). Each niche is thus formed by units connected to similar set of inputs. In the first layer, just one input is allowed; therefore niches are formed by units connected to the same feature. After several epochs of GA with deterministic crowding, the best individual (unit) from each niche is selected to survive in the layer of the model. The construction of the model goes on with the next layers, where niching is also important.

Proposed method of feature ranking patterns on a base principle of NGA monitoring. The basic idea was introduced in [60] and takes into account two factors. First factor is the significance of feature for modelling the output variable. The second one is the amount of additional information to the information carried by already selected variables. This

resembles to the state-of-the-art methods based on a mutual information analysis. These methods select set of features of the highest mutual information with the output variable while minimizing mutual information among the selected features. Kordik [60] found out that by monitoring which genes exist in the population one can estimate the significance of each feature. However, the estimation was highly influenced by the stochastic process, which the genetic algorithms are based on. In next sections of this chapter, we introduce how to overcome this problem and propose a technique called FeRaNGA, which showed great results and an application potential described later in chapter 6.

3.2 Significance estimation

In the initial population of the first layer, units are randomly generated. Connection to certain feature is represented as "1" in corresponding gene locus. Numbers of ones in locus are therefore uniformly distributed at the beginning of GA. After several epochs of GA with DC, numbers of ones in gene loci representing more important features increase, whereas numbers in loci of least significant features decrease (see Fig. 3.1.).

This fact can be used for the estimation of features significance. In each layer of the network, after the last epoch of GA with DC and just before the best gene from each niche is selected, we count how many genes (units) are connected to each input variable. This number is accumulated for each feature and when divided by sum of accumulated numbers for all features, we get the proportional significance of each feature.

3.3 FeRaNGA algorithm for feature ranking

The ranking of features can be extracted from their proportional significance estimated by the above described procedure. This is what we call Feature Ranking utilizing information from Niching Genetic Algorithm (FeRaNGA) algorithm. The configuration of the GAME engine, particularly size of the population and number of epochs of Niching Genetic Algorithm (NGA) has considerable influence on results of the FeRaNGA algorithm. For very low number of epochs, significance of features is close to a random number because the niching genetic algorithm is unable to eliminate units connected to irrelevant features from the population. The GAME engine typically constructs an ensemble of models [78]. We found out that by applying FeRaNGA to all ensemble models and computing the me-

dian from estimated significance of features greatly improve the results. We refer to this method as FeRaNGA-n where n is the number of ensemble models.

3.4 FeRaNGA-n method

The results in section 6 of the FeRaNGA algorithm applied to synthetic data in table 6.1 showed that some redundant features received higher ranking than expected. This behavior was caused by insufficient selection pressure in the niching genetic algorithm - number of epochs was too low to eliminate all redundant features. However, the GAME algorithm usually generates more models for one purpose (ensemble of models). The idea of FeRaNGA-n algorithm is to improve results of ranking by combining unstable FeRaNGA ranks from n GAME models. Final ranks of features are computed as median ranking of features from n models. Results of the FeRaNGA-n method are then very accurate (for more details please see the section 6).

3.5 Conclusion

FeRaNGA-n algorithm is one of the contributions of this thesis. It can be used for feature ranking as well as for feature selection. The advantage of the FeRaNGA algorithm is that it uses the GAME ANN which was designed for data with continuous output variable as well as for categorical variables. Hence, it is able to provide results for both, the classification problem and regression problem, respectively. FeRaNGA does not require any additional computation, all information for ranking are extracted from process of data mining (GAME) models evolution. This approach simplifies the whole process of data mining for researchers from different areas of science as they do not need to have expert knowledge of data mining (especially of methods for feature ranking and feature selection). The only thing they have to is to create the data mining model and then they get all the results at once: i.e. subset of the most important features, ranks for all selected features and results of classification or regression already influenced by the feature ranking and selection results.

Chapter 4

Genetic Programming based Feature Ranking and Feature Selection

4.1 Motivation

In the past two decades, methods of FR, FS and FE were researched from several different approaches. In this section, we describe a new Genetic Programming based approach for FR and FS, [14, 15, 70], which is designed to provide not only the FR and FS results, but also a more complex solution of data mining.

GP was introduced 30 years ago [14, 15, 70], but the main ideas to evolve computer programs as well as representation of the evolved program by the tree structure remain unchanged. These are crucial to our approach because there are no inside loops and all possible program trees describe correct programs syntactically.

Rather than relying on a significant amount of task-specific knowledge, evolutionary methods require only a credit assignment mechanism that is representation-specific rather than task-specific. Poli [70] noted that a representation-specific method of credit assignment avoids the need for explicit task knowledge. The lack of a need for explicit task knowledge plays an important role in the design of the complex problem-specific solution. **Since the GP evolved program can also be an expression, we proposed to use this expression as a scoring function for FR.**

The GP technique in FR, FS and FE methods has mainly been used to generate new features from the original dataset (FE). This task is performed without prior knowledge of

the probabilistic distribution and demonstrates the superior searching power of GP over other techniques [43, 44, 79, 80]. By using GP, brain tumors were classified with FS by [81] as well as it was successfully applied for classifier construction in [45]. Smith and Otero [82, 79] combined GP and GA to perform feature generation and FS with a C4.5 classifier. Lin [83] constructed a classifier of capability for FS and FE using a layered GP. Nesthatin [84] used GP to evolve decision stumps and ranked features by how often particular features were utilized in the process of stump creation in conjunction with the fitness. Ahmed [85] studied the ability of GP with two existing embedded FS metrics for automatic selection of important features and for classification of the selected subset, so the classification accuracy was also an output of this GP-based algorithm. In contrast to [85], we use GP to evolve a feature scoring function directly by using the GP.

The goal of this chapter is to describe a novel problem-adaptive algorithm for feature ranking and feature selection for classification. As we already stated, the problem-adaptive means that the proposed algorithm does not have a predefined measure for FR or FS before a run of the evolutionary algorithm. This measure (scoring function) is discovered during the GP evolution by evaluation on the given dataset and is therefore adapted to the solved problem.

4.2 Description of the algorithm

In this section, we present a description of our algorithm. This algorithm encapsulates two new methods: genetic programing-based feature ranking (GPFR) and genetic programming-based feature selection (GPFS). The algorithm automatically adapts a feature scoring function to a given data set, assigns ranks to all input features (GPFR) and selects a subset of the most relevant features (GPFS). The feature scoring function, a cornerstone of our algorithm, is evolved by GP and represents a mathematical expression describing importance of each input feature. A general scheme of the algorithm is in Figure 4.1.

This figure is devided into two parts. The upper half shows a basic GP schema of the algorithm and the lower part describes evaluation of the feature scoring function quality. The GPFR fitness function consists of two main steps. In the first step, the importance of each input feature is computed by a scoring function. Thus, the ranks to all of the input features are assigned. In the second step, a fitness value of the ranking given by the scoring

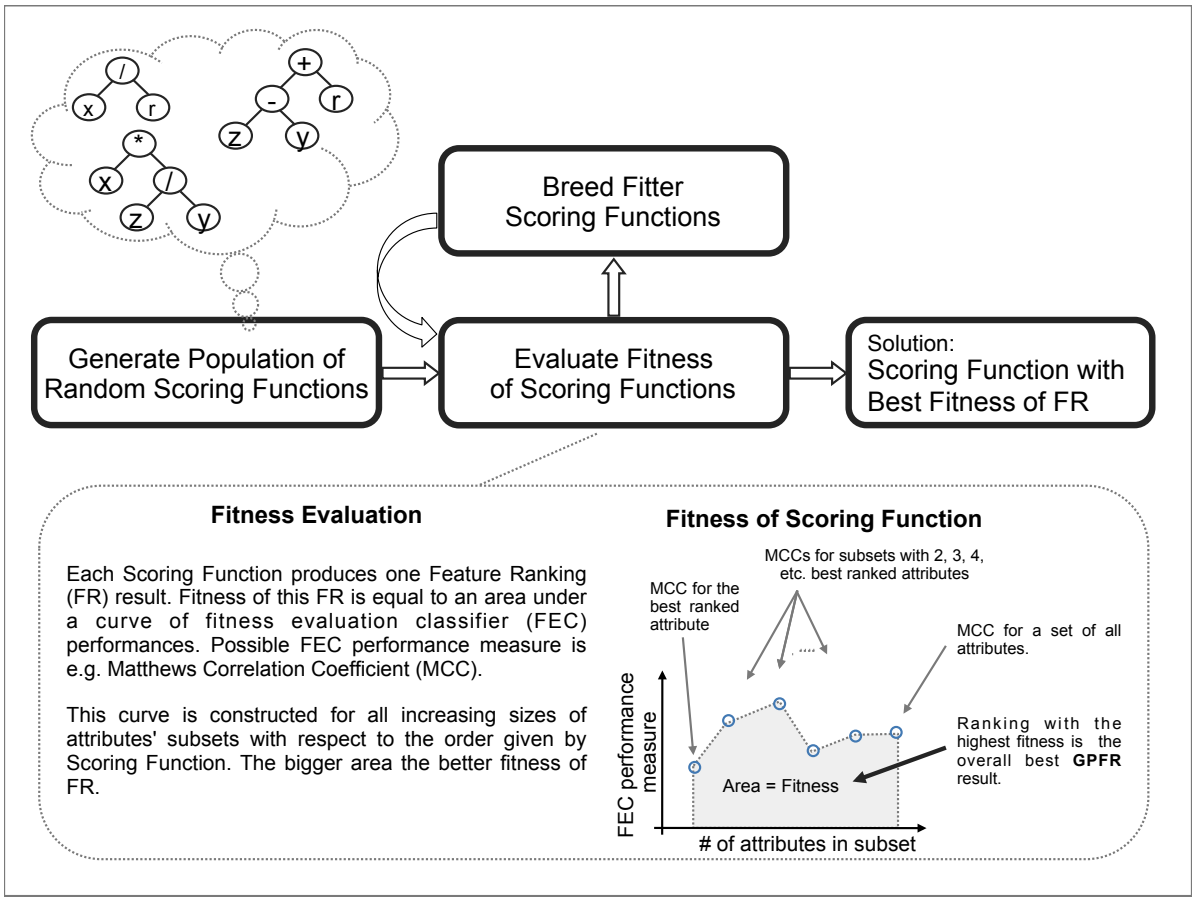

Figure 4.1: GPFR scheme. The upper half of the figure shows the basic GPFR process. The lower part describes a process of fitness evaluation and final results determination.

function is computed by using of fitness evaluation classifier (FEC). Detailed description of the fitness and its evaluation process is described in a subsection 4.2.1. The basic steps of this GP-based algorithm are summarized in pseudo-code in Algorithm 1.

In general, we have created only one algorithm which is able to provide several data mining results at once. **The main goal of the algorithm is to evolve a feature scoring function and produce the best ranking of all input features** at the end of the algorithm. For defining what the best ranking is and what the feature scoring function consists of we have to provide a definition of the fundamentals of our GP algorithm. These fundamentals (theoreticaly described in chapter 2.5) are defined as follows:

- **Terminal set:** $\{C, sd_f, r_{fc}, sd_c, r_{avg}\}$.
- **Function set** consists of $\{+, -, *, /\}$, unary minus, square root.

Data: input data set
Result: ranking and importances of input features, evolved feature scoring function, best feature subset cardinality
Randomly create an initial population of scoring functions from the available primitives;
repeat
  Execute each scoring function in a population and compute its fitness.;
  Select the best scoring functions from the population with a probability based on fitness to participate in genetic operations;
  Create new individual scoring function(s) by applying genetic operation;
until *an acceptable solution is found or a maximum number of generations is reached*;

Algorithm 1: Pseudo code of the GPFR algorithm

- **Fitness measure:** fitness of the scoring function is an area under the curve of classification accuracies (This is a crucial piece of this approach and is described in detail later in this section and showed in Fig. 4.1).

- **Result and termination criterion:** Feature scoring function that maximizes the area under the curve of classification accuracies (provides the best feature ranking result) and FS cardinality.

The terminal set used in our GP implementation contains several statistical characteristics of the data set. The C is a random constant, sd_f is the standard deviation of a scored feature, r_{fc} is the correlation between a scored feature and class feature, sd_c represents the standard deviation of a class feature, r_{avg} is measure of average correlation of a scored feature to all other features (except the class feature). The function set contains basic mathematical functions including unary minus and a square root as recomended in standard in GP. The division by zero and square root of -1 are protected and returns 1 in both cases.

4.2.1 Fitness measure

The **fitness** measure is a crucial part of the GP algorithm. It has to be clear what the goal of the algorithm is. Then it is important to define how to evaluate individuals in population and whether the goal was reached or not. The fitness of an individual represents how the individual is close to the optimal solution. In our case, the goal of GP is to evolve a feature scoring function. To define a fitness of the feature scoring function means to

define a quality of feature ranking result (done by the feature scoring function). In case of synthetic data, we know how the ranking of all features should look like; therefore it would not be a problem to define the fitness as a simple check between two ordered lists of features. Unfortunately, in real world we have to deal with real-world data sets where the ideal ranking is not known and we have to define different fitness of the ranking. A quality of different feature rankings can be compared using data mining model performance (e.g. classification accuracy (CA) or Matthews correlation Coefficient (MCC) in case of classifier or root mean square error (RMSE) in case of regression model). To be able to compute a quality of feature ranking result as one number, we have defined the fitness for classification task and CA as a performance measure as follows:

- **The fitness of an individual is computed as the area under the curve of CAs computed on a subset of features that are defined by rank assignment. The number of subsets to be evaluated is equal to the number of input attributes. The subsets are created as follows. The first subset contains only the best ranked feature. The second subset contains the two best ranked features, and this continues to the last subset which contains all of the features. The CAs are computed by a fitness evaluation classifier (FEC).**

For regression tasks, the FEC can be replaced by any regression model and the fitness is computed as the smallest area under the curve of RMSEs. The reason for this definition of fitness was our hypothesis that the best order of all ranked features has the greatest area under the curve subsets' model performances. Graphical schema of the fitness has been already showed in Figure 4.1. A main advantage of the external classifier used in the process of fitness computation is a problem customization. If the classification problem is being solved, the appropriate (or best or default) classifier is used. On the other hand, if the regression problem is being solved, the classifier can be easily replaced by appropriate regression model.

The ranking produced by the feature scoring function is one of the GPFR method results. The second GPFR result is a mathematical expression describing the relationship between input features and output feature. Third output of the GPFR method is the importance of each feature - the result of the feature scoring function. As we can see from the Figure 4.1, the best classification accuracy is identified during a process of fitness evaluation (across all the individuals and generations of the GP algorithm). We denoted this result

as Genetic Programming based Feature Selection (GPFS), because the number of features determined as a best feature subset providing the best classification accuracy is usually smaller than the overall number of features in data set. This has been already proved for example by Kohavi in [7], where he stated that a significant improvement in accuracy is achieved for some datasets with decision trees. Based on that study and due to the course of dimensionality, we have decided to start with decision tree [86] as a classifier to evaluate the fitness of GPFR in the first exploration in this area.

4.3 Feature selection in GPFR

As we mentioned in section 2.3 the cardinality k of a subset is either prespecified (k-parametrized search) or is allowed by be optimized in the course of search (k-optimizing). The GPFR algorithm can be used in both variants. In the and also the k-parametrized version it selects best k features and this k is also used for the size of subset in fitness evaluation. The k-optimizing version selects such k from the best GP individual of the GPFR run, which maximizes the model performance criteria (e.g. max(CA)). The k-optimizin variant is called GPFS.

4.4 Implementation details and settings

In the following list, GP objective and settings for GPFR algorithm for initial experiments is summarized. According to [70] it is impossible to make general recommendations for setting optimal parameter values, as these depend too much on the details of the application. However, genetic programming is in practice robust, and it is likely that many different parameter values will work. As a consequence, one need not typically spend a long time tuning GP for it to work adequately. In the case of GPFR, the terminal set, function set, and other important values of GP parameters were for initial experiments set to standard values listed in generally respected John Koza's books [16, 87].

- **Terminal set:** $\{C, sd_f, r_{fc}, sd_c, r_{avg}\}$.
- **Function set:** $\{+, -, *, /\}$, unary minus, square root.
- Number of individuals in population: 1000

- Number of epochs: 50
- Ramped half-and-half initialization with a depth range of: 2-6
- Elitism: 2
- Selection by Tournament of size: 7
- Probability of crossover point: 0.9 for functions, 0.1 for terminals
- Max depth of crossover: 5
- subtree crossover: 0.9
- subtree mutation: 0.01

4.5 Conclusion

The GPFR and GPFS methods are the second contribution of this thesis. We have focused on a Genetic Programming based approach for FR and FS, which is designed to provide not only the FR and FS results, but also a complex problem-specific solution of data mining problem. This solution covers a data mining model as well as the results of feature selection and feature ranking already applied in the process of the data mining model building phase. Another advantage of this GP-based approach is the evolved mathematical expression that describes the importance of input attributes. The results of this approach were successfully verified and compared in section 7. Further advantage of this methodology is its ability to solve not only the classification task but also the regression task and is therefore universally applicable.

Part I

Experiments and results

Chapter 5

Evaluation Overview and Data Sets Description

In this chapter of the thesis, a brief overview of techniques used for experiments preparation and results evaluation and comparison is presented. Further, the description of synthetic and real-world data sets used for the comparison is given at the end of this chapter.

5.1 Experiments Preparation

This section introduces the data preparation processes and techniques that we used for computation of results through our proposed methods and on the given data sets. First of all, the data balancing technique used for the results computation is discussed. Then, the description of subsampling strategy to evaluate stability and robustness of FS algorithms follows.

5.1.1 Data balancing

In case of classification task, where classification categories are not equally distributed, the data set has to be corrected, e.g. if the class variable includes 39 negatives versus 170 positives. Hence, such an imbalanced dataset must be corrected [88]. To reduce the different distributions of the training and testing datasets and to minimize the influence of the imbalanced dataset on the correct results, we applied a data balancing technique

(Figure 5.1). To avoid a situation in which the training or testing dataset contains only one type of cases (true or false), we decided to keep an approximately equal ratio of the different cases across all of the created training and testing datasets.

Moreover, to eliminate a problem of overfitting or loss of information and to deal with balancing of training and testing data sets, we applied the data balancing showed in Figure 5.1. To avoid a situation when training or testing dataset contains only one type (true or false) of cases, we decided to keep an approximately equal ratio of different cases across all created training and testing datasets.

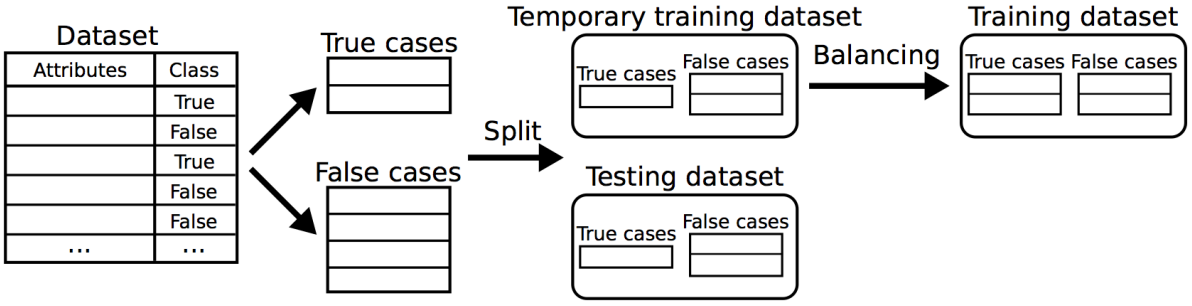

Figure 5.1: Data balancing. Prior to balancing, a dataset was divided into two groups: true and false cases. These two groups were proportionally divided into a temporary training dataset and a testing dataset with approximately equal ratios of true and false cases across all of the datasets. The temporary training dataset was balanced (not reduced) to contain an equal number of true and false cases. Due to the relatively high ratio of true and false cases and the potential influence of the balanced testing dataset on final results, the task of balancing was performed only on the training dataset.

Prior to balancing, a dataset was divided into two groups: true and false cases. These two groups were proportionally divided into a temporary training dataset and a testing dataset with approximately equal ratios of true and false cases across all of the datasets. The temporary training dataset was balanced (not reduced) to contain an equal number of true and false cases. Due to the relatively high ratio of true and false cases and the potential influence of the balanced testing dataset on the final results, for the oral exostoses dataset the task of balancing was performed only on the training dataset.

5.1.2 K-fold cross-validation

Because of the stability and robustness of results, the K-fold cross-validation [6] can be used. A common practise is to used 10-fold cross-validation and repeate the process X times

on a data set that was randomly mixed before balancing. The setup of the experiments with a total of 100 validations can be used as a subsampling strategy to evaluate the stability and robustness of FS algorithms [6].

5.2 Evaluation Overview

In this section, an evaluation overview is presented. There are two techniques described: one for several FR and FS results combination and the second technique for the results evaluation and interpretation.

5.2.1 Combining of multiple results

Under certain circumstances (low stability), it is possible to get several results from one or several FR or FS methods. To be able to interpret these results (and have only one ranking of all data attributes), it is possible to combine these results together.

The most common method is the Borda count (BC) method [89]. This preferential voting method, which can be classified as single winner voting, can also be considered as a generalization [90]. We prepared 5 variants of BC, which are called bc_1, bc_2, bc_3, bc_4 and bc_5. The BC formula for a particular feature (we denote this general formula as bc_1) is defined as follows:

$$bc_1 = N - r \tag{5.1}$$

where N is the total number of features and r is the final rank of a feature. The features is one particular feature from the set of GPFR or GPFS results. The formulas for the other three BCs are defined as

$$bc_2 = \frac{N - r}{N} \tag{5.2}$$

$$bc_3 = \frac{1}{r} \tag{5.3}$$

$$bc_4 = \frac{w * (N - r)}{N} \tag{5.4}$$

$$bc_5 = w * (N - r) \tag{5.5}$$

where w represents the weight applied to a feature, N is the total number of features, and r is the ranking of a feature. In GPFR, N is equal to all of the features in the dataset. In contrast to bc_1, bc_2 reflects a different number of selected attributes among the datasets and is chosen mainly to combine the FS results with different numbers of selected features, whereas bc_3 greatly penalizes bad ranks, so better features are preferred. The weighted bc_4 reflects a different number of selected attributes among the datasets as well as a weight that is assigned to a candidate. bc_5 is similar to bc_4, but it does not reflect a number of selected features and is therefore more appropriate for the FR results combination. In terms of FS, the weight is the importance of a feature that is assigned by a scoring function.

5.2.2 FR and FS results evaluation

For FR and FS results evalution a several measures based on the confusion matrix were used. They are namely: Matthews correlation coeficient (MCC), area under ROC curve (AUC ROC), F-score and classification accuracy (CA). Detailed overview can be found e.g. in [91] and [92].

The usage of ROC AUC has been questioned in classifier results comparison, because the AUC can be quite noisy [93]. However, this should not affect our comparison of results obtained for several methods on the same classifier.

5.3 Data Sets

In this thesis, we have used synthetic data sets as well as real-world problem data sets. The synthetic data sets are introduced first, then the real-world data sets.

5.3.1 Synthetic Data Sets

Both synthetic data sets are describing classification problems. In the first case, it is a Gaussian Multivariate data set, the second synthetic dataset is Uniform Hypercube data set.

5.3.1.1 Gaussian Multivariate data Set

This artificial data set consists of two clusters of points generated from two different 10th-dimensional normal Gaussian distributions and was created by M. Tesmer and P. A. Estevez for experiments in [56]. Class 1 corresponds to points generated from N(0, 1) for each dimension and Class 2 to points generated from N(4, 1). This data set consists of 50 features and 500 samples per class. By construction, features 1 to 10 are equally relevant, features 11 to 20 are completely irrelevant and features 21 to 50 are highly redundant with the first ten features. Ideally, the order should be as follows: at first relevant features 1 to 10, then the redundant features 21 to 50, and finally the irrelevant features 11 to 20.

5.3.1.2 Uniform Hypercube Data Set

Second artificial data set consists of two clusters of points generated from two different 10th-dimensional hypercube $[0, 1]10$, with uniform distribution. The relevant feature vector (f1, f2, ..., f10) was generated from this hypercube in decreasing order of relevance from feature 1 to 10. A parameter $\alpha = 0.5$ was defined for the relevance of the first feature and a factor $\alpha = 0.8$ for decreasing the relevance of each feature. A pattern belongs to Class 1 if ($f_i < \gamma^{i-1} * \alpha$ / $i = 1, \ldots, 10$); otherwise, it belongs to Class 2. This data set consists of 50 features and 500 samples per class. By construction, features 1 to 10 are relevant, features 11 to 20 are completely irrelevant, and features 21 to 50 are highly redundant with first 10 features. Ideally, the order should be as follows: at first relevant features 1-10 (starting with feature 1 until feature 10 in the last position), then the redundant features 21 to 50, and finally the irrelevant features 11 to 20. This data set also comes from [56].

5.3.2 Real-world data sets

In this subsection, three real-world problem data sets are described. The Ionosphere, Boston Housing and QSAR biodegradation data sets were used as a benchmark data sets

commonly used in a field of machine learning. These data sets are freely available from UCI machine learning repository [94]. In this thesis, we have also used two specific real-world data sets from a field of anthropology and dental medicine. The first biomedical data set is used for oral exostosis modeling and the second one is used for dental age estimation, both are described in detail later in this subsection 5.3.2.4 and 5.3.2.5

5.3.2.1 Ionosphere real-world data set

This radar data (from ML UCI repository [94]) was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an auto-correlation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal. Number of instances is 351; number of attributes is 34 and there is one class attribute. All predictor attributes are continuous.

5.3.2.2 Housing real-world data set

This Boston Housing Data set (from ML UCI repository [94]) was taken from the StatLib library which is maintained at Carnegie Mellon University. This data set is multivariate and consists of 506 instances and 13 attributes. All attributes are continues and there is no missing value. Following list of attributes is important, because during experiments we often work with attribute index only, thus attribute number 1 represents the CRIM attribute in this listing. Attribute Information:

1. CRIM per capita crime rate by town
2. ZN proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS proportion of non-retail business acres per town
4. CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)

5. NOX nitric oxides concentration (parts per 10 million)
6. RM average number of rooms per dwelling
7. AGE proportion of owner-occupied units built prior to 1940
8. DIS weighted distances to five Boston employment centres
9. RAD index of accessibility to radial highways
10. TAX full-value property-tax rate per
11. PTRATIO pupil-teacher ratio by town
12. B proportion of blacks by town coefficient
13. LSTAT - lower status of the population
14. MEDV Median value of owner-occupied homes

5.3.2.3 QSAR Biodegradation Real-World Data Set

This QSAR biodegradation data set (available at ML UCI repository [94]) has been used to develop QSAR (Quantitative Structure Activity Relationships) models for a study of relationships between chemical structure and biodegradation of molecules. Biodegradation experimental values of 1055 chemicals were collected from the webpage of the National Institute of Technology and Evaluation of Japan (NITE). Classification models were developed in order to discriminate ready (356) and not ready (699) biodegradable molecules by means of three different modelling methods: k Nearest Neighbours, Partial Least Squares Discriminant Analysis and Support Vector Machines. The data set consists of 41 molecular descriptors and 1 experimental class. Number of instances of this multivariate data set is 1055 and represents a classification task. All attributes are integer or real values. There is no missing value in this dataset.

5.3.2.4 Dental Age real-world data set

Anhropologists solve often a problem of age estimation. One possibility is to utilize a dental age estimation, which is based on a tooth mineralization process. Unfortunately, this approach works only on children, because adults have the tooth mineralization process

already finished. This data set consists of sample of 1393 orthopantomographs taken of 657 boys and 736 girls of Czech nationality aged between 3 and 17 years (i.e., children who had reached 3 and still under 18 years). Panoramic radiographs were performed on individuals treated at three clinics in Prague, Czech Republic (Stomatology Clinic of the Faculty Hospital Královské Vinohrady, Pediatric Stomatology Clinic of the Faculty Hospital Motol and Stomatology Clinic of the First Faculty Hospital in Prague). Imaging was carried out mainly in the 1990s (range from 1981 to 2007). The radiographs were digitized and classified between 2005 and 2008. The selected children were born from 1973 to 2004. They were healthy, without any growth disorders. Individuals with severe orthodontic defects and individuals of other nationality were excluded from the study. All the children were of known chronological age (in decimal years; from birth to the time the radiograph was taken). Each observation (one row) in the dataset consists of 16 input data attributes (all real values) and one output attribute. The input attributes represents left and right half of upper jaw (8 attributes each half). Each attribute contains a real value describing a state of the mineralization of the particular tooth. According to Moorrrees-Fanning-Hunt classification scheme (MFH, [95]) the developmental stages of all mandibular teeth were identified and transformed into numbers as follows: Initial cusp formation was denoted as stage 0, the Coalescence of cusps as stage 1 and so forth until the last stage Apical closure complete denoted as stage 14. There were also some inter-stages, e.g. a value 12.5 can appear. There were missing values in the data set. We discovered that the appropriate method for missing value replacement is either to take the mirror tooth value (if exists) or to take an interpolation of the nearest older and younger patient's value. Output attribute represents age of the patient. The age is a real value computed as a year plus proportional part of months of a patient.

5.3.2.5 Oral Exostoses real-world data set

This data set has been collected from the anatomical collection of the Chiang Mai Faculty of Medicine, Chiang Mai, Thailand, during February 2012 by a sole trained observer. The collection only comprises of Thai individuals originating from Chiang Mai or its immediate surroundings. We sampled in that collection 209 individuals (103 females, 106 males), aged 15 to 96 (mean age: 66.68 years, standard deviation: 15,45). In order to determine and classify which parameters may participate in the apparition of oral exostoses, various parameters were recorded:

- V1-V6: oral exostoses classes representing presence of particular oral exostosis;
- V7-V24: cranial variants;
- V25-V73B: dental variants;
- V74-V125: data on oral health status, including the evaluation of carious lesions, periodontal disease, dental wear and tooth displacement;
- V126-V134: data on occlusal relationships;
- V135-V143: occlusal measurements;
- V144-V161B: craniofacial measurements and stature.

Cranial and dental variants have been chosen as genetic proxies. Oral health status, occlusal relationships, occlusal measurements and craniofacial measurements represent environmental proxies. More detailed overview of all input attributes is listed in publication [A.2]. The attributes V1 to V6 are binary class attributes. There are 5 types of oral exostoses. The class attribute V1 represents presence of any of the 5 types of oral exostoses. Class attributes V2 to V6 represents presents of a particular exostose. Based on medical experts decision, this dataset was divided into 6 separate datasets consisting of one dependent class attribute (V1-V6) and the rest of all attributes. Thus, we have here 6 exostoses dataset named Exostoses1 (with class attribute V1), Exostoses2 (with class attribute V2) and so on up to Exostoses6 (with class attribute V6).

5.3.2.6 Oral Exostoses real-world data set

MADELON is an artificial dataset (available at ML UCI repository [94]) containing data points grouped in 32 clusters placed on the vertices of a five dimensional hypercube and randomly labeled +1 or -1. The five dimensions constitute 5 informative features. 15 linear combinations of those features were added to form a set of 20 (redundant) informative features. Based on those 20 features one must separate the examples into the 2 classes (corresponding to the +-1 labels). Probes are the random features.

Number of variables/features/attributes: Real: 20 Probes: 480 Total: 500

Chapter 6

Results of the FeRaNGA-n Method

In this chapter, the results of the FeRaNGA and FeRaNGA-n methods are presented. The chapter is divided into two main parts. In the first part, section 6.1, experimental analysis of the method is performed an evaluated, and then the method is compared the most common statistical methods for feature ranking and feature selection 6.1.2. In the second section, FeRaNGA-n method is applied to a real world problem of age modeling from teeth mineralization.

6.1 Results of FeRaNGA and FeRaNGA-n method

In this section, we compared several statistical methods for feature ranking with new method for feature ranking derived from data mining process - FeRaNGA. The comparison is performed on both artificial and real-world data sets. We compared the performance of FS algorithms available in WEKA on synthetic data sets 5.3.1.2 and 5.3.1.1 generated by Tesmer and Estevez to measure the performance of the AMIFS method [56]. We adjust their performance to improve results on synthetic data sets. Finally, we applied all FS algorithms to real-world data set. Data sets used for the second part of the comparison are real world Housing data set, Ionosphere data set and QSAR biodegradation data set already described in chapter 5.

6.1.1 Experimental Analysis

A lot of experiments with the FeRaNGA and FeRaNGA-n methods were performed to show behavior under different circumstances and to find optimal settings to provide best results. The first experiment, 6.1.1.2, compares the performance of feature ranking methods in their implicit configuration on three different data sets. The GAME used standard configuration with 15 individuals in the NGA and 30 epochs. We run the algorithm several times to show the unstable behavior. The second part of analysis describe experiments 6.1.1.3 with making the FeRaNGA algorithm more restrictive to prevent irrelevant features being selected by a chance. Further, the results of the FeRaNGA-n algorithm with well configured GAME engine are presented 6.1.1.4. There are also shown results of experiments with different parameters setings 6.1.1.5 and finally, results comparing rankings from different layers of the ANN network 6.1.1.6.

6.1.1.1 Feature ranking methods of the state-of-the-art selected for comparison

Seven Weka FR methods partially described already in chapter 2.3.5, in more detail in [18] was used for comparison of ranking performance. ChiSquared method evaluates the worth of an attribute by computing the value of the chi-squared statistic with respect to the class, GainRatio by measuring the gain ratio with respect to the class, InfoGain by measuring the information gain with respect to the class, OneR by using the OneR classifier. ReliefF evaluates the worth of an attribute by repeatedly sampling an instance and considering the value of the given attribute for the nearest instance of the same and different class. SVM evaluates the worth of an attribute by using an SVM classifier and SymmetricalUncert(SU) evaluates the worth of an attribute by measuring the symmetrical uncertainty with respect to the class.

6.1.1.2 FeRaNGA algorithm vs WEKA FR methods in default settings

Most of FR methods in Weka are giving exact results corresponding to artificial data sets characteristic, except SVM (as is shown in Table 6.1 and 6.2). In these tables only the interesting columns are displayed (it means features, which were ranked and selected by FR and FS algorithms). The reference ranking of features is displayed in first rows. A light grey backgrounded cells with black coloured numbers represent features which have zero

significance (no units connected to them were able to survive the evolutionary process) and dark grey with white coloured numbers represents redundant features.

Table 6.1: Ranks of features for Gaussian Multivariate Data Set used in WEKA and GAME with default settings.

Method	Relevant(1-10)										Redundant(21–50) and Irrelevant(11–20)																					
ChiSq	7	6	9	10	8	2	1	3	5	4	38	26	21	48	50	33	29	40	49	27	22	42	32	34	30	17	16	20	15
GainRatio	10	6	5	9	7	1	3	2	4	8	38	49	27	47	29	36	33	22	37	48	24	39	43	28	34	18	20	19	17
InfoGain	7	6	8	9	2	10	1	3	5	4	38	26	29	48	50	40	21	33	46	27	24	22	31	32	43	17	16	20	15
OneR	7	9	10	8	2	3	1	6	4	5	38	26	25	40	29	28	50	21	24	33	35	43	30	34	27	16	11	12	14
ReliefF	6	9	7	4	10	3	5	2	1	8	26	36	27	50	21	46	30	49	28	31	23	45	41	47	48	18	15	16	13
SVM	2	4	6	3	7	9	1	10	8	23	47	39	30	37	42	22	34	21	38	44	5	17	41	14	28	20	45	18	25
SU	10	6	7	9	5	1	3	2	4	8	38	47	49	27	29	33	36	48	37	22	24	44	39	40	30	18	20	19	17
GAME 1	7	8	2	6	26	23	9	35	28	4	1	33	38	40	43	20	42	47	3	5	15	21	22	24	34	36	37	39	48
GAME 2	10	7	3	9	37	27	24	50	8	5	23	46	44	1	2	4	6	11	12	13	19	28	29	30	36	38	39	40	47
GAME 3	10	2	5	8	7	6	3	29	26	4	37	47	36	46	16	34	43	1	9	11	18	23	24	25	32	33	35	38	48
GAME 4	3	9	10	41	6	7	5	8	27	25	31	48	16	39	20	34	44	28	35	1	14	21	22	23	32	33	36	37	46
GAME 5	9	6	3	10	28	50	4	38	29	8	13	36	48	27	20	15	18	23	42	1	14	22	24	25	34	37	39	40	46

Black background cells with white numbers are irrelevant features and cells with white background are relevant. This colour separation is valid for all tables except Table 6.3 where a real world dataset is used and there is no prior knowledge about relevance or irrelevance of attributes.

Results of the FeRaNGA method are presented in 5 independent runs of the GAME DM algorithm denoted as GAME and followed by index of the algorithm run. For the Gaussian data set, the order of the first ten selected features is not important, because all 10 features are equally significant. WEKA's methods (except the SVM) ranked features correctly (see Table 6.1).

The FeRaNGA algorithm demonstrated worse results in comparison with WEKA's methods. Due to randomness of niching genetic algorithm used and insufficient number of epochs, ranks are different for each GAME model. Here we wanted to present that the FeRaNGA ranking algorithm is not working well without experimental analysis of its behavior.

Table 6.2 shows more or less similar results. All methods from WEKA ranked first ten features correctly, except ReliefF and SVM methods. Results of the FeRaNGA method are unstable, except the first position. The most significant feature was identified correctly for all 5 GAME models.

Table 6.2: Uniform Hypercube Data Set analysed with default settings of WEKA and GAME methods.

Method	1	2	3	4	5	6	7	8	9	10	Redundant(21-50) and Irrelevant(11-20)																						
ChiSquare	1	2	3	4	5	6	7	8	9	10	22	37	48	35	39	43	47	26	45	27	30	49	29	32	28	33	46	31	42	17	18	14	
GainRatio	1	2	3	4	5	6	7	8	9	10	23	38	27	35	37	30	34	39	28	25	22	26	50	32	36	21	31	46	33	17	18	14	
InfoGain	1	2	3	4	5	6	7	8	9	10	22	37	48	43	35	39	26	47	44	30	27	49	36	29	33	28	23	46	42	17	18	14	
OneR	1	2	3	4	5	6	7	8	9	10	48	37	43	50	40	30	25	44	39	45	22	49	38	34	36	33	23	28	24	18	17	14	
ReliefF	1	2	3	4	8	7	9	10	5	6	29	48	47	25	31	30	49	26	27	28	21	50	24	43	45	42	23	20	44	15	41	17	
SVM	1	2	48	3	29	43	4	9	35	7	36	45	41	44	34	25	8	11	12	30	37	19	15	16	6	38	24	20	5	13	40	28	
SU	1	2	3	4	5	6	7	8	9	10	23	38	35	37	27	48	39	34	26	25	50	45	49	29	32	21	31	46	24	17	18	14	
GAME 1	1	2	4	3	5	27	49	45	6	47	32	7	8	33	35	38	24	43	13	16	21	23	37	14	28	40	9	12	18	20	29	50	
GAME 2	1	2	3	4	44	38	5	48	25	7	22	40	32	19	47	50	20	46	31	35	43	18	37	8	9	10	12	14	16	21	30	49	
GAME 3	1	2	3	5	7	47	37	4	48	6	8	26	30	43	45	13	14	18	42	9	10	11	12	17	19	20	23	25	29	32	38	50	
GAME 4	1	2	3	33	4	30	41	7	44	29	46	47	8	5	9	25	34	36	48	10	35	40	6	13	14	16	18	20	22	24	31	50	
GAME 5	1	6	3	2	27	8	50	4	5	48	7	22	25	28	39	47	9	16	29	30	31	11	13	40	46	49	10	15	20	26	35	45	

In the Table 6.3 we can see results for real-world data, Housing Data Set. This time, even methods from WEKA differ in their ranking. It indicates that this problem is more complex than synthetic data. All methods found first feature (Criminality in the area) as the most significant feature for the value of housing in Boston. When the average ranking is computed from WEKA methods and from several GAME models, results are very similar.

Table 6.3: Housing data set - results from GAME vs Weka. In columns, there are ordered features according to their ranking. Most significant features are on the left. Average rankings were computed from obtained feature importances.

Method	Features Significance													Method	Features Significance											
ChiSquare	1	6	4	7	5	2	8	10	11	3	9	12		GAME 1	1	6	7	4	5	10	9	2	3	11	8	12
GainRatio	1	10	3	2	7	12	5	11	9	8	4	6		GAME 2	1	7	6	4	2	8	9	10	11	3	12	5
InfoGain	1	4	6	7	2	5	8	10	11	3	9	12		GAME 3	1	7	4	6	3	9	8	2	10	5	12	11
OneR	1	7	6	5	4	2	11	8	3	10	9	12		GAME 4	1	6	7	5	2	12	10	9	4	8	11	3
ReliefF	1	7	6	4	11	5	9	3	12	10	8	2		GAME 5	1	7	6	8	5	9	10	2	4	3	11	12
SVM	1	6	2	7	4	12	3	8	5	11	9	10		-	-	-	-	-	-	-	-	-	-	-	-	-
SU	1	4	2	6	7	5	8	10	3	11	9	12		-	-	-	-	-	-	-	-	-	-	-	-	-
Average	1	7	6	4	2	5	3	10	8	11	12	9		**Average**	1	7	6	4	2	9	5	10	8	3	12	11

6.1.1.3 Restricted FeRaNGA algorithm - more selective ranking

The results of the FeRaNGA algorithm applied to synthetic data (Table 6.1) showed that some redundant features received higher ranking than expected. This behavior is caused by insufficient selection pressure in the niching genetic algorithm - number of epochs was too low to eliminate all redundant features.

In this section, we have experimented with inner settings of the FeRaNGA algorithm. The ranking is computed from best chromosomes (feature lists) from the population. Number

of unique chromosomes (UNC) used for the ranking varied from 1 to all (Table 6.4).

Table 6.4: Restricted FeRaNGA algorithm on the Hypercube data set. The bigger number of UNC causes the bigger number of used features.

UNC	1	2	3	4	5	6	7	8	9	10	Redundant(21–50) and Irrelevant(11–20)														
All	1	2	3	4	5	6	46	15	24	37	48	13	17	21	28	29	30	41	43	45	49	50	7	8	9
1 / 2	1	2	3	4	26	12	35	46	5	6	7	8	9	10	11	13	14	15	16	17	18	19	20	21	22
1 / 3	1	2	37	21	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	22	23	24
1 / 4	1	2	41	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Results were measured on Hypercube Data Set with following settings of NGA: number of epochs 150 and size of initial population 150. It can be observed, that when UNC is lower, number of used features is also reduced. For very low values of UNC only relevant features are ranked, while for all UNC redundant and irrelevant features are ranked as well. Detailed overview for different FeRaNGA-n settings are showed in table 6.8.

The FeRaNGA algorithm restricted to one UNC (the best chromosome from the population) can be used to find and rank just few most important features of the data set. In the next section we present results of the FeRaNGA-n algorithm which is powered by ensemble methods.

6.1.1.4 Results for FeRaNGA-n algorithm

The GAME algorithm usually generates more models for one purpose: to provide an ensemble of models. The idea of FeRaNGA-n algorithm is to improve results of ranking by combining unstable FeRaNGA ranks from n GAME models. Final ranks of features are computed as median ranking of features from n models. In the Tab. 6.5 we showed results of the FeRaNGA-5 algorithm on Hypercube data set. Restrictive trend corresponding with number of UNC is again evident. NGA configuration was 150 epochs and size of initial population as well.

Results of the FeRaNGA-n are very accurate. All selected features are relevant and have correct ranks. For all UNC only 7 from 10 relevant features is selected (last row in Tab. 6.5), but their ranks are accordant with their real ranks. To change the number of selected features, we can reconfigure the NGA as shown in next section.

Table 6.5: Results of FeRaNGA-5 algorithm on Hypercube data set. All selected features are relevant and have correct ranks.

UNC	1	2	3	4	5	6	7	8	9	10	Redundant (21 – 50) and Irrelevant (11 – 20)													
2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1 / 4	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1 / 3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1 / 2	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
2 / 3	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
All	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24

6.1.1.5 Parameters of NGA for FR

The performance of the FeRaNGA-n algorithm on Hypercube data can be improved by reconfiguration of NGA parameters. The table 6.6 presents results of FeRaNGA-7 algo-

Table 6.6: Different configuration of FeRaNGA-7 on Hypercube data set.

Config.	1	2	3	4	5	6	7	8	9	10	Redundant(21-50) and Irrelevant(11-20)													
15-30	1	2	3	4	6	7	8	5	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
30-30	1	2	3	4	5	6	7	8	10	39	48	9	11	12	13	14	15	16	17	18	19	20	21	22	23
50-30	1	2	4	3	5	7	6	8	9	22	37	10	11	12	13	14	15	16	17	18	19	20	21	23	24
100-30	1	2	3	4	5	7	6	8	9	10	22	26	34	35	37	38	40	44	45	47	48	11	12	13	14
150-30	1	2	4	3	5	6	8	7	22	9	34	35	26	39	43	44	10	30	40	45	47	48	11	12	13
300-30	1	2	3	4	5	7	6	8	9	43	37	22	39	48	44	26	27	38	50	34	35	36	10	11	12

rithm with all UNCs and NGA's configurations displayed in the first column. First number in column Configuration represents size of the initial population and the second number stands for the number of epochs.

When ranks for more (or for all) features are needed, reconfiguration of parameters of NGA can take place. However, there is no guarantee that all the features will be selected by NGA and thus ranked by FeRaNGA simply because irrelevant or redundant features are not supposed to be selected in NGA. Results in table 6.6 are not quite accurate because of low number of epochs. An improvement can be achieved by increased number of epochs or number of GAME models from which medians are chosen. In Table 6.7 results of FeRaNGA-5 for different number of epochs are displayed. There is obvious trend that the higher number of epochs has significant influence on final ranks.

Results for different combination of UNC settings and configuration of NGA on Hypercube

Table 6.7: Different configuration of NGA. In this experiment all unique chromosomes were used in FeRaNGA-5 algorithm on Hypercube data set. Accuracy of final ranks is improved by increasing number of epochs of the NGA.

Configuration \ Ideal ranking	1	2	3	4	5	6	7	8	9	10
30 30	1	2	3	4	5	6	8	7	37	9
50 50	2	1	4	3	5	6	7	8	22	48
75 75	1	2	3	4	5	6	7	8	9	10
100100	1	2	3	4	5	6	7	8	9	10
150150	1	2	3	4	5	6	7	8	9	10

data set are displayed in Table 6.8. Configuration of the GAME algorithm consists of number of epochs (first number) and size of initial population. Grey backgrounded features are redundant. The bigger number of UNC causes the bigger number of used features. For lower number of epochs and individuals in NGA population there are some features that have a wrong position than they are supposed to have, e.g. for UNC = 1/4 and 50 50 NGA configuration. For better configurations is this defect eliminated.

Results for the same settings of UNC and configuration of NGA on Gaussian data set are shown in Table 6.9

6.1.1.6 Different ranks among layers of the GAME network

Table 6.10 shows an example how the ranking looks like when more than one layer is observed. From the results it is quite clear, that results only from the first layer are the best one across all layers and configurations of the NGA. This statement will be confirmed by all result in this section below.

Grey background means again not selected attribute. Ranks with blue background should be by definition placed on the left side, the dark red marked ranks on the opposite site, fully on the right side and the light-red ranks in the middle. There is obvious, that the lower number of the layer reaches the better ranking. The best rankings are provided from the first layer (denoted as 0).

Very important was to compare ranking results among all created layers of the GAME model identified by FeRaNGA-n method and computed for different number of models n. Results of these experiments for Gaussian data set are shown in Table 6.11.

Configuration of the NGA was 30 individuals in population and 15 epochs. Results are

Table 6.8: Restricted ranking provided by FeRaNGA algorithm on the Hypercube data set for several combinations of different number of unique chromosomes (UNC) and different configuration. There are empty cells, because there were no more selected features. The bigger number of UNC causes the bigger number of used features. Configuration of the GAME algorithm consists of number of epochs (first number) and size of initial population. Grey backgrounded features are redundant. For lower number of epochs and individuals in NGA population there are some features that have a wrong position than they are supposed to have, e.g. for UNC = 1/4 and 50 50 NGA configuration. For better configurations is this defect eliminated.

UNC	Configuration	FeRaNGA-5 ranks								
2	30 30	1	2	3						
	50 50	1	2							
	75 75	1	2							
	100100	1	2							
	150150	1	2							
3	30 30	1	2	4						
	50 50	1	2	3	4					
	75 75	1	2	3						
	100100	1	2	3						
	150150	1	2	3						
1/4	30 30	1	3							
	50 50	2	1							
	75 75	1	2	3						
	100100	1	2							
	150150	1	2							
1/3	30 30	1	4	3	2	5				
	50 50	1	3	2	5					
	75 75	1	2	3						
	100100	1	2	4	3					
	150150	1	2	3						
1/2	30 30	1	2	7	3	4	6			
	50 50	1	3	2	4	5				
	75 75	1	2	3	4	5				
	100100	1	2	3	4	5				
	150150	1	2	3	4	5				
2/3	30 30	2	1	4	6	3	5	7		
	50 50	1	2	3	4	5	6			
	75 75	1	2	3	5	4	6	7		
	100100	1	2	3	4	5	7	6		
	150150	1	2	3	4					
ALL	30 30	1	2	3	4	5	6	8	7	37
	50 50	2	1	4	3	5	6	7	8	22	48
	75 75	1	2	3	4	5	6	7	8	9	
	100100	1	2	3	4	5	6	7	8	9	
	150150	1	2	3	4	5	6	7			

Table 6.9: Restricted FeRaNGA algorithm on the Gaussian data set for several combinations of UNC and different configuration. Only selected features by NGA were ranked. There are empty cells are because there were no more selected features. The bigger number of unique chromosomes (UNC) causes again the bigger number of used features. Configuration of the GAME algorithm consists of number of epochs (first number) and size of initial population. Grey backgrounded features are redundant.

UNC	Configuration	FeRaNGA-5 ranks											
3	30 30	7											
	50 50	6	7										
	75 75	7	3	6									
	100100	9	2										
	150150	10	7										
1/4	30 30	10	3										
	50 50	7	5										
	75 75	7	9										
	100100	10											
	150150	8	6										
1/3	30 30	8	9	6	1								
	50 50	6	10										
	75 75	7	5	6									
	100100	6	10	1									
	150150	10	9	6									
1/2	30 30	6	8	9	7	10	46						
	50 50	9	6	3	7	10	2	8					
	75 75	7	8	9	10	6	2	1	5				
	100100	7	9	10	6	8	2	3					
	150150	7	6	8	9								
ALL	30 30	5	6	8	10	1	2	3	4	7	9	22	33
	50 50	8	6	2	7	10	3	5	9	1	4	26	
	75 75	3	7	6	9	8	2	1	5	10	4	38	
	100100	8	6	7	10	2	9	3	5	1	4		
	150150	7	9	8	6	10	2	3	1	5	4		

Table 6.10: Different GA configuration of GAME for FeRaNGA-7 used to display ranks from all generated layers of the GAME ANN for the Hypercube data set. Grey background means again not selected attribute. Ranks with blue background should be by definition placed on the left side, the dark red marked ranks on the opposite site, fully on the right side and the light-red ranks in the middle. There is obvious, that the lower number of the layer reaches the better ranking. In the first layer there are all attributes that are important for the problem and there is very limited number of redundant attributes selected.

Configuration	Layer	Ranks - FeRaNGA-7																	
50 50	0	1	3	4	2	5	6	7	8	9	10	28	48	11	12	13	14	15	16
	1	1	2	3	4	7	5	6	8	9	10	28	48	49	16	41	44	45	46
	2	1	7	2	3	4	5	6	8	48	20	49	46	9	10	28	16	19	37
50 100	0	2	1	3	4	8	5	7	6	9	10	11	12	13	14	15	16	17	18
	1	1	2	3	4	8	5	7	6	9	25	28	30	34	39	49	50	10	11
50 150	0	1	2	3	4	5	6	8	7	9	10	11	12	13	14	15	16	17	18
	1	1	2	3	6	4	5	15	8	32	45	46	7	9	10	11	12	13	14
50 200	0	1	3	2	4	6	5	7	8	9	10	11	12	13	14	15	16	17	18
	1	1	3	2	4	6	47	5	7	17	26	32	33	44	11	21	22	25	29
	2	1	3	2	48	4	17	7	6	5	47	26	43	32	38	11	33	41	44
50 250	0	1	2	3	4	6	5	7	8	9	10	11	12	13	14	15	16	17	18
	1	1	2	3	4	6	5	23	26	7	16	18	22	27	37	43	8	9	10
	2	1	5	2	3	6	4	7	22	43	23	26	28	39	21	37	36	16	18

shown for all layers generated by the GAME algorithm. Rows denoted as overall represent ranks computed from all the generated layers of the model. Features with blue background are equally relevant and in ideal case they should be placed on the first ten positions from the left. If there are empty cells, the models simply did not selected more features, thus the ranks were not computed for them. Cells with black background contain irrelevant features.

All other numbers with white background represent redundant features. It is obvious, that the lower number of models is used for ranking, then the higher number of irrelevant features occurs. The irrelevant features occur only in higher layers of the GAME network. It is caused by definition of the NGA that also less important features are allowed to participate on the building process of the model.

The higher number of models used for ranks computations tend to better results of ranking. First layer of each FeRaNGA-n run shows the best performance. Ranks in second and other layers are not as accurate as in first layer. The higher number of models used for ranking the better results of ranking is obtained and also the lower number of redundant and irrelevant attributes is selected.

When the same experiment was repeated for NGA settings 75 (75 individuals in population and 75 epochs), the results were better for the configuration 75 than for the previous configuration (30 individuals and 15 epochs). However, there were for higher layers still

Table 6.11: Results of FeRaNGA-n algorithm on Gaussian data set for different number of models with NGA configuration 30 individuals in population and 15 epochs. Relevant features have a blue background. Results are shown for all layers generated by the GAME algorithm. Rows denoted as overall represent ranks computed from all the generated layers of the model. The higher number of models used for ranks computations tend to better results of ranking. First layer of each FeRaNGA-n run shows the best performance. Ranks in second and other layers are not as accurate as in first layer. The higher number of models used for ranking the better results of ranking is obtained and also the lower number of redundant and irrelevant attributes is selected.

	Layer	Ideal ranking: random combination of 1 to 10, 21 to 50, 11 to 21																		
FeRaNGA-1	0	7	9	5	2	4	8	25	37											
	1	7	9	5	8	48	2	4	25	37	21	26	31	42						
	2	7	9	5	48	8	2	31	4	25	37	20	21	26	42	46	12	29	41	
	Overall	7	9	5	8	48	2	4	25	37	31	21	26	42						
FeRaNGA-2	0	9	7	3	4	5	1	2	6	8	10	25	27	37	38					
	1	9	7	6	3	5	4	8	48	1	27	2	10	25	37	38	50	12	21	22
	2	9	7	6	3	5	48	4	2	8	50	27	1	31	38	10	25	37	12	42
	Overall	9	7	6	3	5	4	8	48	2	1	27	10	25	37	38	50	12	31	42
FeRaNGA-3	0	9	4	5	2	6	7	10												
	1	9	5	4	2	3	6	7	10											
	2	9	5	3	7	4	2	6	10	8	12	14	28	37	42	26	31	41	47	
	Overall	9	5	4	7	2	3	6	10											
FeRaNGA-4	0	9	4	5	6	7	10	1	2	28	38									
	1	9	7	5	4	10	6	2	1	3	28	38	22	25	31					
	2	9	5	7	3	10	4	2	6	31	1	28	33	25	38	12	8	22	14	20
	Overall	9	5	7	4	10	6	2	1	3	28	38	25	31	22					
FeRaNGA-5	0	9	4	5	6	7	10													
	1	9	7	5	4	3	6	10	1	2										
	2	9	7	3	5	4	6	10	2	38	1	20	28	37	26	31	41	46	12	
	Overall	9	7	5	4	3	6	10	2	1										
FeRaNGA-6	0	9	3	4	5	6	7	10	1	2										
	1	9	7	6	3	4	10	5	1	2	25	28								
	2	9	7	3	5	4	6	10	2	38	1	20	28	37	26	31	41	46	12	
	Overall	9	3	7	6	4	10	5	1	2	28									
FeRaNGA-7	0	9	1	2	3	4	5	6	7	10										
	1	9	10	5	6	7	4	1	2	3										
	2	9	7	3	5	10	4	6	2	1	31	8	28	38	46	20	37	26	29	41
	Overall	9	5	6	7	10	3	4	1	2										

Table 6.12: Results of FeRaNGA-n algorithm on Gaussian data set for different number of models with NGA configuration 150 individuals in population and 150 epochs. Relevant features have a blue background. Results are shown for all layers generated by the GAME algorithm. Rows denoted as overall represent ranks computed from all the generated layers of the model. All the ranks are correct and there is no selected irrelevant feature. In all layers 10 or more features are selected, in layer 0 exactly 10 features.

	Layer	Ideal ranking: random combination of 1 to 10, 21 to 50, 11 to 21																			
FeRaNGA-1	0	2	7	10	9	3	8	5	6	1	4										
	1	2	7	10	9	3	8	5	6	4	1	29	32	39	28	30	34	45			
	Overall	2	7	10	9	3	8	5	6	4	1	29	32	39	28	30	34	45			
FeRaNGA-2	0	7	10	2	9	1	6	8	3	5	4										
	1	7	2	10	1	8	9	3	6	5	4	26	22	24	29	32	37	38	39	50	45
	Overall	7	10	2	1	9	8	6	3	5	4	26	22	24	29	32	37	38	39	50	45
FeRaNGA-3	0	10	7	9	8	3	2	5	6	1	4										
	1	10	7	9	3	8	2	5	6	1	4	26	22	24	29	32	37	45			
	Overall	10	7	9	8	3	2	5	6	1	4	26	22	24	29	32	37	45			
FeRaNGA-4	0	7	10	9	2	3	8	6	5	1	4										
	1	7	10	2	3	8	9	5	1	6	4	22	25	26	37	24	27	29	50	45	30
	Overall	7	10	2	3	9	8	5	6	1	4	22	25	26	37	24	27	29	50	45	30
FeRaNGA-5	0	7	10	2	9	6	3	8	5	1	4										
	1	7	2	10	3	8	1	9	5	6	4	25	22	27	29	35	37	45			
	Overall	7	2	10	3	9	8	6	5	1	4	25	22	27	29	35	37	45			
FeRaNGA-6	0	7	2	10	9	6	8	3	5	1	4										
	1	2	7	8	9	10	3	6	5	1	4	25	22	45	23	27	28	29	35	37	44
	Overall	2	7	10	9	8	6	3	5	1	4	25	22	45	23	27	28	29	35	37	44
FeRaNGA-7	0	7	10	2	9	6	3	8	5	1	4										
	1	2	7	10	8	6	3	9	5	1	4	25	22	28	37	44	45				
	Overall	7	2	10	9	6	3	8	5	1	4	25	22	28	37	44	45				

some redundant features ranked higher than the relevant ones and also some irrelevant features occurred. Nevertheless, all ranks in the first layer were correct and there were only 9 or 10 features selected.

The best results for Gaussian data set were achieved for the NGA configuration 150 (150 individuals in population and 150 epochs) as shown in Table 6.12.

The same experiment was repeated also for Hypercube data set and for the same configurations of the NGA. Results were very similar to the experiments for Gaussian data set, but the improvement among configurations was not as good as for Gaussian data set. However, the ranks in first layer were also correct all the time. Table A.1 shows the results for configuration of NGA 150 individuals in population and 150 epochs.

Results in higher layers are not so clear as for Gaussian data set and for number of model between 4 and 7 there are some irrelevant features selected. As the increased number of models to be used did not helped we assume, that there are limitations of the FeRaNGA-n method in higher number of layers which cannot be easily overcome by different settings of the method. Results for Hypercube data set with settings 150 individuals in population and 150 epochs for FeRaNGA-14 are shown in appendix in Table A.2. Ranks in first layer are still correct and the number of selected features varies from 6 to 9.

6.1.2 Comparison on Real-World Data Sets

. In this subsection, the FeRaNGA-7 method is compared with several feature ranking and feature selection methods from state-of-the-art. The real-world data sets described in chapter 5 were used. Classification accuracy was computed by the Random Forest [86] classifier using 10 times repeated 10-fold cross-validation technique. The results of the classification are displayed in following Figures and Tables as a median values from these 10 runs of the evaluation. Results stated in this subsection were not published yet.

The first data set used for the comparison was the Ionosphere data set. Following feature ranking and feature selection methods were used for comparison: CFS (correlation based feature selection subset evaluator with BestFirst search method), ReliefF and InfoGain (Information Gain. Both subset evaluators with Ranker search method). The settings of these methods was not changed and used as set by default in WEKA. The FeRaNGA-7 method was used with the following NGA settings: number of individuals in population 150 and number of epochs also 150.

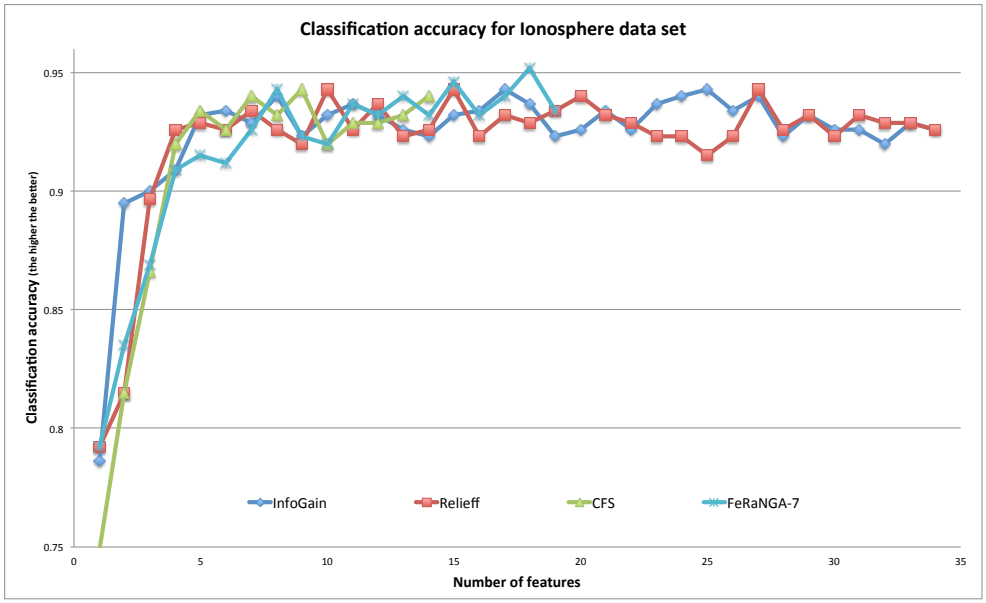

Figure 6.1: Classification accuracy for the Ionosphere data set - comparison of the FeRaNGA-7 method with CFS, ReliefF and InfoGain methods. The FeRaNGA-7 ranks were obtained for the NGA settings of 150 individuals in population and 150 of epochs. All methods provided similar and comparable result, however, the best classification accuracy was achieved by the FeRaNGA-7 method for 18 features selected in the best subset.

Results on the Ionosphere data set are showed in Fig. 6.1. Different number of ranked features is given by the definition of the CFS (feature selection method) and the FeRaNGA method (embedded feature ranking and feature selection method). All the methods provided high classification accuracy (more than 90% for 4 and more features), however, the FeRaNGA method achieved the overall best classification accuracy 95,2%. The number of selected features used when the maximal CA was reached, is 18 for FeRaNGA-7.

We have performed other statistical evaluation of this FeRaNGA-7 result on Ionosphere data set as well and also other measures (MCC, F-measure and AUC ROC correspond with the CAs results (showed in 6.1) and they supported above described result of the FeRaNGA-n method.

The second comparison of the FeRaNGA method was performed on QSAR biodegradation data set, again the classification accuracy was used. Feature ranking and feature selection methods remain the same as in previous experiment. Results are shown in Figure6.2. Features were ordered for CA computation according to their assigned importance (rank) generated from the particular method. The most important feature was on the left. The

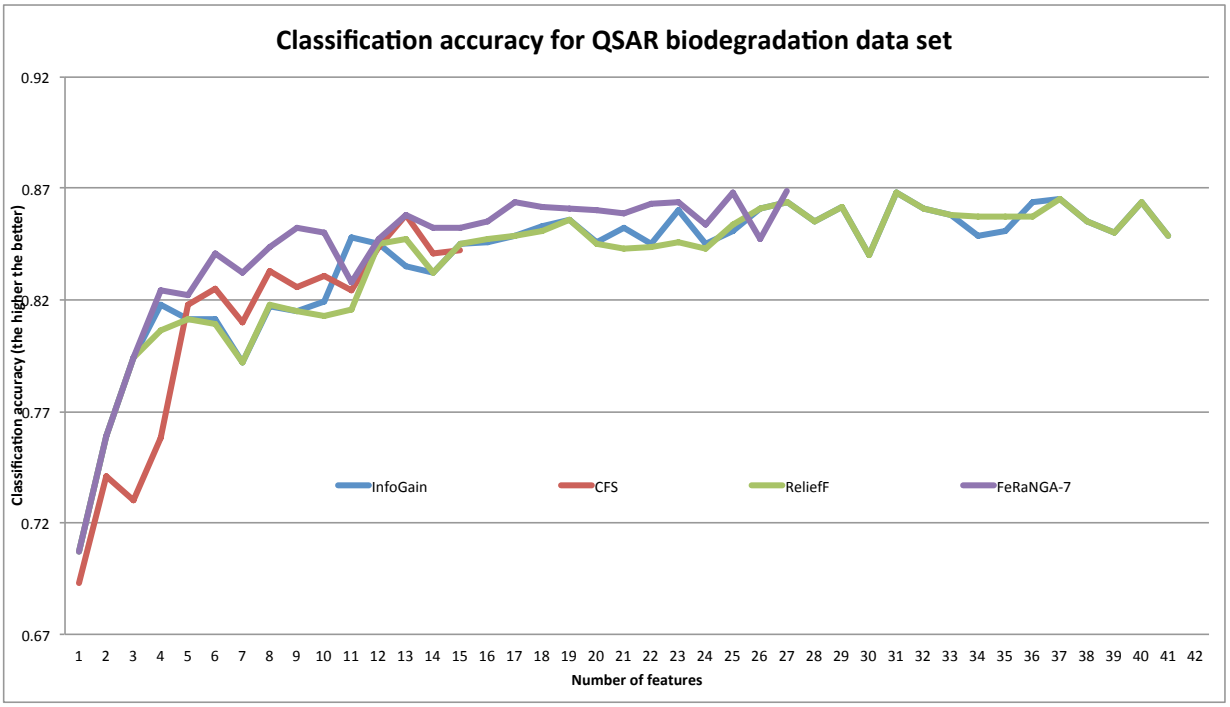

Figure 6.2: Results of classification accuracy for FeRaNGA-n method on QSAR biodegradation data set in comparison with several state-of-the-art methods results. FeRaNGA-n results were obtained for NGA settings of 150 individuals in population and 150 epochs. Attributes are ordered from the best one on the left to the worst one on the right. The best classification accuracy (CA) is achieved for the FeRaNGA-n method.

best classification accuracy of the FeRaNGA-7 method is obvious.

For the Biodegradative dataset we have performed the same evaluation as for the Ionosphere dataset and obtained results clarifying the CA results described in Figure 6.2. These results are located in appendix: the Matthews correlation coefficient in Figure A.1.

6.1.3 Conclusions

In general, it is possible to utilize the GAME ANN for feature ranking and feature selection and to obtain at least comparable or in many cases better results. The FeRaNGA-n algorithm for FR works well not only on artificial data sets but also for the real world data set. The comparison of FeRaNGA-n algorithm and FR methods from WEKA showed, that results on Hypercube data set are (from the definition of correct ranking) equivalent, but the number of selected features depends on the configuration of NGA and on the number

of GAME models from which the median ranks are chosen by FeRaNGA method. Proper settings of the algorithm were investigated and it is obvious, that:

- the best rankings are provided by FeRaNGA-n from the first layer independently of the NGA algorithm settings
- adequate number of GAME model to be generated should be at least 5
- when a higher number of ranked features is requested, parameters of the NGA should be reconfigured
- from the definition of the FeRaNGA-n method, it cannot provide rankings for all attributes, because of the selection process in the GAME building phase.

FeRaNGA-n algorithm can be used for FR as well as for feature selection. The advantage of the algorithm is that it can be used for data with continuous output variable as well as for categorical variables. It does not require any additional computation, all information for ranking are extracted from process of data mining (GAME) models evolution. Therefore data miners can build the data mining model and get all results at once (classification accuracy or error of the model, subset of the most important features and ranks and importances for the selected features. This advantage allows them to use our approach as one "black box" method and to focus to other parts of their research in different discipline to apply or analyse the obtained results.

6.2 Application of the FeRaNGA-n method in age estimation from teeth mineralization

In this section, the FeRaNGA-n method is applied for real world problem where an age is estimated from teeth mineralization and the most important teeth during the prediction need to be identified. This research was published in impacted journal [96]. Dental development is frequently used to estimate age in many anthropological specializations. The aim of this section is to show how useful can be the FeRaNGA-n approach in finding of accurate predictive age system for the Czech population. It can help to discover any different predictive ability of various tooth types and their ontogenetic stability during infancy and adolescence. The accuracy of dental age estimation decreases with increasing age. While age prediction in children is sometimes given in months, in adults it can often be determined only in decades.

6.2.1 Data set and experiment design

This experiment was performed on the Dental Age real-world data set, which was already described in section 5.3.2.4. In the first part of this experiment the whole dataset was used. For the second part, the data set was divided into three groups. The age groups (3-7.99, 8-12.99, 13-17.99 years) were created to see whether the accuracy of dental age estimation and the significance of teeth in a predictive model were consistent during ontogenetic development.

The data set was divided at random into a training part for model construction and a testing part to validate the constructed model. We used the same training and testing parts for all methods to obtain comparable results. Moreover, we could compare Feature Selection and Feature Ranking results as well as ages predicted for a given individual by different modeling methods. The testing part has the actual chronological age recorded and this allows us to compare the predicted and actual age and thus assess model accuracy.

6.2.2 Data Mining methods used for comparison

For comparison with the GAME ANN where the FeRaNGA algorithm is incorporated, we decided to use simple linear regression (SLR) and Multiple Linear regression (MLR),

Figure 6.3: X-ray image of a patient. Similar X-ray images were used for individual teeth mineralization classification. Source: author of this thesis.

the Support Vector Machine (SVM) and finally the Radial Basis Function (RBF) neural network. Detailed description of all method can be found in [18]. To express the accuracy of DMM, we used the Root Mean Square (RMS) Error. RMS indicates the average difference between actual chronological age and age estimated from teeth development. The more accurate the estimate of a model, the lower the value of RMS. Ideally, the predicted age is the same as the chronological age and the individuals contribution to RMS is zero. All the presented RMS errors are calculated on the testing set.

6.2.3 Results

In the first part of this section, we compared the results of various methodological testing approaches. The second part describes significance of tooth types identified by FeRaNGA method in age estimation.

6.2.3.1 Results of dental age estimation

The results of various methodological testing approaches, some of which are commonly used for dental age estimation (simple linear regression, linear regression, SVM prediction, RBF neural network, GAME neural network). Average RMS errors for age estimation are calculated separately for girls and boys, including their 10th and 90th percentile, as shown in Table 6.13.

Table 6.13: Results of tested multivariate methods expressed by average RMS errors separately for girls and boys, including their 10th and 90th percentiles.

Sex	**Girls**			**Boys**		
Method	RMS	10^{th} perc.	90^{th} perc.	RMS	10^{th} perc.	90^{th} perc.
Simple regression	1.27	–1.78	1.63	1.38	–1.48	1.33
Linear regression	1.00	–1.38	1.23	1.11	–1.37	1.27
SVM prediction	0.99	–1.52	1.17	1.13	–1.41	1.10
RBF neural network	1.12	–1.52	1.32	1.08	–1.15	1.12
GAME neural network	0.93	–1.06	1.01	1.07	–1.13	1.20

Table 6.13 shows that with the exception of the simple regression method, the results of the tested multivariate methods were essentially comparable (differences in the order of hundredths of a year). If we focused on estimating the interval between the 10th and 90th percentile, the most accurate method was the GAME neural network in girls (2-year interval) and both artificial neural network methods in boys (2.3-year interval). The simple regression method predicted the age from only one tooth; therefore, it was expected to be the least accurate this was confirmed (3-year interval in girls). The reason for the slightly better performance of the GAME and RBF methods is that they consist of a larger number of units, permitting each unit to solve a small portion of the problem. In this way, they can handle local variances with higher accuracy. In contrast, the SVM and both regression methods solve the problem globally.

In appendix there are Figures A.2 and A.2, which describe the chronological age of 220 girls and 170 boys randomly selected for the testing part, including their age estimation using the most successful predictive models. Graphical comparison shows that, for younger children, all the methods tend to predict a higher age than the actual one. This is evident for the RBF artificial neural network. In contrast to girls (Fig. 2), boys (Fig. 3) show a higher variance in the predicted age and therefore higher RMS error.

6.2.3.2 Significance of tooth types identified by FeRaNGA method in age estimation

Significances of various permanent tooth types in age estimation were obtained using the FeRaNGA-7 method with following configuration of the niching genetic algorithm: 150 individuals in population and 150 epochs of the algorithm. The values in Tables 6.14, 6.15,

6.16 and 6.17 are the significances of a given tooth identified by FeRaNGA-7 method, thus a median from 7 models is selected. If the value is missing, the model has not used the tooth. The significance of each tooth in age estimation was computed both collectively (3-17.99 years), as well as in three sub-ages (3-7.99, 8-12.99, 13-17.99 years).

Table 6.14: Significance of individual tooth in the predictive model for the age period 3-17 years.

Sex/teeth		I1	I2	C	P1	P2	M1	M2	M3	Mean RMS
Girls	Dx	–	–	–	1.47	0.40	–	22.27	4.00	
	Sin	–	–	–	0.53	0.33	–	70.80	–	**0.93**
Boys	Dx	–	–	–	0.47	0.40	–	60.66	–	
	Sin	–	–	–	–	0.13	–	38.33	–	**1.07**

During the overall period from 3 to 17 years, the predictive ability of second molars was found to be the most significant. The predictive contribution of second molars of both quadrants was the strongest of all the teeth in any sectional age period. The reason was probably that, in this broad age period, second molar mineralization underwent almost all developmental stages. RMS error in the Czech population reached 0.93 year in girls and 1.07 years in boys (Table 6.14). The importance of the second molar in age estimation is further supported by the fact that simple linear regression predicts the age only based on this tooth. Although the predictive significance of other teeth was not high, their influence on the predicted age cannot be neglected. As shown in Table 6.13, methods taking into account other teeth, including linear regression, achieved significantly better results.

Due to random initialization, the GAME model started using either the left or right tooth. And since the other teeth brought no new information, their use was suppressed. In the next repetition, the model could be initialized and vice versa. This is supported by the fact that, in almost all cases, if one tooth was significant, the contralateral tooth had nonzero significance.

In the youngest age category (3-7 years), many sexual differences in dental mineralization were found (Table 6.15). Mineralization of boys teeth was delayed compared to girls teeth, because earlier mineralized teeth had stronger predictive properties in boys. Teeth M1, I2 and I1 had the strongest predictive values in boys. In contrast, M1 had no predictive ability in girls, while the most important tooth was the second premolar (P2). The predictive models for both sexes were consistent only in the fact that third molars did not contribute

Table 6.15: Significance of individual teeth in the predictive model for the age period 3-7 years.

Sex/teeth		I1	I2	C	P1	P2	M1	M2	M3	Mean RMS
Girls	Dx	7.33	5.33	10.00	6.66	14.00	–	4.00	–	
	Sin	8.66	4.67	8.67	9.33	12.00	0.67	8.67	–	**0.52**
Boys	Dx	10.67	17.33	–	0.67	2.00	18.66	2.67	–	
	Sin	8.66	13.33	0.67	6.66	2.00	16.00	0.67	–	**0.62**

to prediction. The contributions of the other permanent teeth were relatively balanced. The smallest RMS error was noted in girls and boys in the lowest age group (3-7 years; 0.52 year in girls, 0.62 year in boys). With increasing age, the predictive accuracy decreased. As shown in Fig. A.3, the ANN ignored boys below 5 years and predicted them to be about 5. But this did not affect the overall accuracy of the model, because there were too few boys of this age and the accurate prediction for older boys outweighed this error.

Table 6.16: Significance of individual teeth in the predictive model for the age period 8-12 years.

Sex/teeth		I1	I2	C	P1	P2	M1	M2	M3	Mean RMS
Girls	Dx	–	–	–	8.67	8.00	–	24.66	–	
	Sin	–	–	–	8.00	9.33	0.67	40.66	–	**0.67**
Boys	Dx	–	–	–	8.67	2.00	12.67	36.66	1.33	
	Sin	–	–	0.67	2.67	–	6.00	24.66	4.67	**0.78**

Sex differences in the predictive significance of individual teeth were less pronounced in the second age period (8-12 years; Table 6.16). The most reliable tooth for age estimation was the second molar (M2); I1, I2 and C had almost no predictive value in both sexes. For male age prediction, M1 and P1 were also applicable. On the other hand, P1 and P2 contributed to age prediction in the female model. RMS deviation errors were 0.67 years in girls and 0.78 years in boys.

For age prediction of both sexes in the last age category (13-17 years), M1, I1, I2 and C were not important (Table 6.17). The significance of other teeth, however, was different for both sexes. The strongest predictive value was found for the third molar (M3), less than P1 and P2 in boys. M2 and P2 and to a lesser extent M3 and P1 contributed the most to

age prediction in the female model. RMS deviation in this last age category reached 1.20 years in girls and 1.22 years in boys (Table 6.17).

Table 6.17: Significance of individual teeth in the predictive model for the age period 13-17 years.

Sex/teeth		I1	I2	C	P1	P2	M1	M2	M3	Mean RMS
Girls	Dx	–	0.67	1.33	3.33	17.33	–	22.67	8.00	
	Sin	–	–	2.00	2.67	16.66	–	15.33	10.00	**1.20**
Boys	Dx	–	3.33	–	12.00	8.00	–	2.00	23.33	
	Sin	–	2.00	–	6.00	5.33	–	1.33	36.66	**1.22**

6.2.4 Conclusion

The aim of this section was to show how powerful and useful is the integrated combination of embedded feature selection (done by the GAME ANN) with the FeRaNGA-n feature ranking method. It allows using maximum information from data sets in general. In case of teeth developmental stages it is about to create an accurate age prediction model for the Czech population. Moreover, the used methodology was also developed to provide valuable information about the predictive importance of particular teeth. From the oral dentistry point of view we have tried to answer the question regarding accuracy of dental age estimation and the predictive importance of teeth changes during the period between 3 and 17 years. In contrast to classical statistical methods, we used data mining methods. Especially the GAME method tends to be more accurate than other used methods, such as simple or multiple linear regressions. Moreover, the FeRaNGA-n method allocates importance to each particular tooth for correct prediction. Based on the importance analysis done by FeRaNGA-n method, we have determined the minimal subset of teeth needed to create an accurate model.

Chapter 7

Results of the GPFS and GPFR methods

The goal of this chapter is to show results and comparison of the second contribution of the thesis - the GPFR (Genetic Programming based Feature Ranking) method. The GPFR settings specified in section 4.4 are kept the same within this chapter with following exceptions:

- Number of individuals in population: 500
- Number of epochs: 20
- Selection by Tournament of size: 3

The reason for the change of settings is related to early experiments with setup of the GPFR evolution described in the section 7.1.

All experiments in this chapter were done using the following validation process (described in [97]), if not stated otherwise. The number of FS trials (size of evaluated system of subsets) was set to 100. From each data set, 1/3 of data in each class was reserved for testing and as such excluded from FS process. In each FS trial, 90% of the remaining data was randomly sampled to form a trial local data set. In the wrapper FS setting, the the criterion value has been obtained as the average over 2 classification rates obtained using 2-fold holdout, where in each loop, the trial-local data had been randomly scattered

to 60 percent training, 30 percent vlaidation, and 10 percent unused data. In the filter setting, the criterion values have been computed from the training data part only. All reported classification rates have been obtained on independent test data. The reason for this validation are following: to avoid over-selection, to minimize GPFR fitness evaluation time and to keep the independent test data.

In wrapper setting, following WEKA DM tool [18] implementations of classifiers were used for the FS criteria computation: Naive Bayes (NB), 3 nearest neghbours (3-NN) and J48 (implementation of C4.5 decision tree). All classifiers had the same default settings as specified WEKA. In the filter setting, implementations of the InformationGain (IG), ReliefF and Support Vector Machine (SVM) methods from WEKA were used in its default settings.

The time performance analyzed in several experiments was done on a machines equiped with: processor 2,2 GHz Intel Core i7, total number of cores 4, memory 16 GB 1600 MHz DDR3.

7.1 Experiments with GPFR evolution setup

Initial experiments with GPFR agorithm were oriented on the setup of the evolution. The objectives of these experiments were to find optimal settings for basic parameters of GP, namely number of epochs and number of individuals in population. The reason for this optimization is the key part of the GPFR algorithm, the fittness function and the number of its evaluation.

The number of fitness evaluations depends directly on the number of epochs and number of individuals in population. Often, to a first approximation, GP runtime can be estimated by the product of: the number of runs R, the number of generations G, the size of the population P, the average size of the programs s and the number of fitness cases F [70]. In case of GPFR, the number of fitness cases is given by number of samples of data we allocated for training. This number differs from hundreds to tens or hundreds of thousends among ML and real-world datasets. This parameter, as well as the number of runs, is not related to GP settings. We did not wanted to modify the size of the programs s, in our case the size of scoring functions, because there is a quite wide range in the default

settings for this size and we do not know in advance how the scoring function should/will look like (exept the already known scoring functions from other feature ranking methods). Therefore, there are only two parameters we can observe and if needed to adjust at the beginnig. At first we used the default settings mentioned in chapter 4.4 and run the GPFR algorithm 100 times with different random seed (25 times on each of 4 datasets of different sizes: Exostoses1, Ionosphere, Hypercube and Madelon). We observed a number of epochs needed to find the best solution of each run. Results of this experiment are presented in table 7.1.

Table 7.1: Number of epochs needed for evolution of the best solution. Mean and standard deviation computed from 100 runs of GPFR for different number of epochs and number of individuals in population.

Settings of epochs in GPFR	50		20	
Number of individuals	1000		500	
Dataset \ Epochs measured	Mean # epochs	SD # epochs	Mean # epochs	SD # epochs
Ionosphere	3,82	2,43	3,93	2,38
Hypercube	1	0	1	0
Exostoses1	5,59	2,9	8,72	4,6
Madelon	2,15	1,71	2,95	2,58

This table contains results for two different settings of GPFR: 50 epochs with 1000 individuals in population and 20 epochs with 500 individuals in population. Mean and standard deviations from 100 runs were calculated. The first experiment with 50 epochs and 1000 individuals showed that maximum mean number of epochs needed to the find best solution was 5.59 for the Exostoses1 dataset. Standard deviations were up to 2.9. If we looked at the results, the overall maximum of epochs was 15 (for the Exostoses1 dataset). Based on this result we decided to lower the settins of maximum number of epochs in GPFR to 20. This settings reduced the runtime approximately to 0.4 of the previous time. Then we

repeated the experiment with this new settings of epochs and we also lowered the number of individuals from 1000 to 500 to see, whether such a huge number of individuals in population is needed. The results of this experiments are also presented in table 7.1. There we can see that the mean number of epochs needed to find the best solution increased for all datasets except the Hypercube dataset. There has to be said, that GPFR found the optimal solution only in case of the hypercube dataset. Here the solution was found within the first epoch and it was needed to evaluate 3.2 of individuals in average over 200 runs. The maximum mean number of epochs needed to find the best solution was again for the

Exostoses1 dataset. If we looked again at the results, the overall maximum of epochs was 19 (for the Exostoses1 dataset). The number of epochs needed to find the best solution was higher than 15 only in 4 cases (3 times for the Exostoses1 dataset and 1 time for the Ionosphere dataset).

In above described experiments we analysed also a diversity of populations. The measure of diversity of populaton was a variety - the number of distinct individuals in the population. This measurement was done among epochs in each particular run of GPFR. It showed that the diveristy was highher at the beginning of the evolution (about 95%) than at the end (75%). The higher number of epochs were done, the lower diversity the population had. We prooved [70, 16] statement that the problem was caused by the number of individuals participating in tournament for selection. The selection preasure was to high. In this case, we lowered the size of tournament from seven to three.

After this change, we repeated the experiment with 100 runs of GPFR with settins: 500 individuals in population, maximum number of epochs 20 and tournament size 3. The diversity of populations were much higher than before (97% at the beginning and 89% at the end of the evolution), but as it is generally known, the higher diversity does not necessarily men the better resuls [70]. In comparison of the results from previous experiments showed in table 7.1, the smaller size of tournament caused only a slightly lower means of epochs needed to find the best solution (changes were up to 0.3 with almost the same standard deviations).

7.2 Test data sample size experiment

In this experiment, the GPFR method was tested on different sample sizes to examine a sensitivity on the testing sample size. Three data sets (Exostoses1, Madelon, Ionosphere) were analized using three different test data sizes: 1/3, 1/2 and 2/3. Other validation settings was the same as specified at the beginning of this chapter. The feature stability was also analyzed in this experiment using the same setup. Results are shown in table 7.2.

The highest criterion values are obtained for the smallest test data, whre only 1/3 was used for independent validation. Nevertheless, the differences for Madelon and Ipnosphere data sets are small, therefore also a 1/2 of the original data size could by used as a test

Data	Train / Test	Crit. Value		Classif. Rate		ASM
		Mean	S.Dv.	Mean	S.Dv.	
Ionosphere	33/66	.88	.143	.85	.073	.75
	50/50	.95	.093	.91	.028	.79
	66/33	.94	.085	.92	.030	.78
Exostoses1	33/66	.85	.137	.76	.732	.55
	50/50	.94	.081	.83	.420	.81
	66/33	.93	.074	.87	.031	.84
Madelon	33/66	.821	.055	.793	.031	.74
	50/50	.835	.028	.806	.029	.81
	66/33	.849	.026	.812	.023	.92

Table 7.2: Test sample size experiments on Exostoses1, Madelon and Ionosphere data sets. The GPFR Stability computed using the ASM measure.

samle. The biggest improvement on test data was achieved on Madelon data set. The ASM stability measure showed also a growing trend with decreasing size of test data and highest ASM value was achieved for Madelon data set.

7.3 Comparison on Oral Exostoses1 and 2 data sets

In this section, two Oral Exostoses data sets were used to compare performance of GPFR to other wrapper methods implemented in the WEKA DM tool. The following search methods were used: Best First search (BF), Genetic Search (GS), Linear Forward Selection LFS, RankSearch (RS) and Subset Size Forward Selection (SSFS) in its default settings.

Results are shown in following table.

Table 7.3 shows comparison of GPFR to other FS wrapper methods on Exostoses1 and Exostoses2 data sets. Last row in each block of results for particular classifier displays also results for a Borda Count combination (bc_2 variant) of all 100 FS trials of the GPFR method's results, here denoted as GPFR-BC. For the Exostoses1 data the GPFR method provided the best results among all wrapper methods using all three classifiers. The highest classification accuracy (CA) is achieved by GPFR with J48 classifier. It also showed, that overall highest CAs are obtained for all methods if the J48 classifier is used. The highest subset sizes are selected by GS method for both datasets. In comparison of results on both datasets, the Exostoses1 data provides higher CA. The reason is in the difference between the class features. The class feature for Exostoses1 represents presence of any

		Exostoses1 data							Exostoses2 data						
		Crit. Value		Classif. Rate		Subset size		Time	Crit. Value		Classif. Rate		Subset size		Time
Wrap.	FS Meth.	Mean	S.Dv.	Mean	S.Dv.	Mean	S.Dv.	(h:m)	Mean	S.Dv.	Mean	S.Dv.	Mean	S.Dv.	(h:m)
J48	BF	.831	.020	.803	.050	5.1	3.71	1:10	.730	.021	.669	.045	4.7	2.71	1:10
	GS	.829	.031	.793	.079	17.2	8.92	2:27	.791	.032	.753	.040	22.4	5.65	2:27
	LFS	.836	.035	.804	.048	8.3	5.27	0:31	.682	.024	.658	.037	2.1	1.30	0:31
	RS	.827	.028	.793	.039	5.7	3.98	2:03	.691	.012	.662	.041	1.83	0.17	2:03
	SSFS	.846	.023	.815	.084	12.4	6.01	0:28	.674	.030	.674	.057	1.94	0.87	0:28
	GPFR	.893	.024	.865	.031	5.2	4.11	316:01	.764	.036	.728	.043	9.14	3.52	316:01
	GPFR-BC	-	-	.856	.053	4.1	1.89	316:01	-	-	.709	.049	11.2	3.28	316:01
NB	BF	.824	.028	.769	.052	6.2	4.84	0:59	.721	.034	.699	.047	11.3	4.21	0:59
	GS	.842	.012	.779	.038	43.7	16.5	2:25	.739	.023	.707	.087	41.4	12.82	2:25
	LFS	.816	.032	.783	.048	8.3	3.36	0:27	.714	.025	.682	.050	1.4	0.24	0:27
	RS	.822	.047	.752	.068	17.8	9.13	1:58	.673	.018	.612	.062	1.7	0.82	1:58
	SSFS	.831	.035	.808	.092	3.7	2,87	0:25	.691	.034	.682	.053	1.1	0.43	0:25
	GPFR	.836	.027	.821	.054	12.9	7.48	297:31	.715	.045	.689	.088	23.1	4.87	297:31
	GPFR-BC	-	-	.805	.048	10.73	2.42	297:31	-	-	.714	.062	16.8	3.29	297:31
3-NN	BF	.837	.022	.788	.048	4.8	3.54	1:19	.724	.021	.693	.047	3.4	1.32	1:19
	GS	.817	.025	.798	.033	55.3	16.8	2:45	.771	.030	.748	.056	30.7	14.5	2:45
	LFS	.798	.018	.760	.028	8.5	3.72	0:55	.715	.015	.693	.082	3.2	2.01	0:55
	RS	.827	.019	.801	.056	4.3	1.56	2:52	.729	.024	.701	.072	1.7	1.15	2:52
	SSFS	.788	.029	.769	.037	5.2	3.58	0:48	.694	.013	.681	.065	2.8	2.54	0:48
	GPFR	.824	.031	.813	.045	11.17	8.42	302:01	.782	.027	.752	.089	17.3	6.28	302:01
	GPFR-BC	-	-	.798	.072	35.2	5.21	302:01	-	-	.745	.038	21.5	3.82	302:01

Table 7.3: Comparison of FS wrappers and GPFR on Exostoses1 and Exostoses2 datasets.

oral exostosis, while the Exostoses2 class variable represetns only a presence of the oral exostoses of type V2.

The overall best CA on Exostoses2 dataset was achieved by GA method with J48, but without a significant difference to GPFR (the second best result) with 3-NN. Good result also provided the GPFR-BC method with NB and 3-NN and seems to be also promising technique. This Borada Count technique could be used not only for GPFR results combination, but also as a technique combining FS results from all the defined or used methods. This has not been done here, but it should be investigated in future work. When discusing the use of J48, it seems to be so far the best option among wrapper approache for data set Exostoses1 and Exostoses2.

7.4 Comparison on Oral Exostoses3, 4, 5 and 6 data sets

In this section, the rest four Oral Exostoses data sets were used to compare performance of GPFR to the same wrapper methods as mentioned in previous section. Table 7.4 shows

		Exostoses3 data							Exostoses4 data						
		Crit. Value		Classif. Rate		Subset size		Time	Crit. Value		Classif. Rate		Subset size		Time
Wrap.	FS Meth.	Mean	S.Dv.	Mean	S.Dv.	Mean	S.Dv.	(h:m)	Mean	S.Dv.	Mean	S.Dv.	Mean	S.Dv.	(h:m)
J48	BF	.782	.023	.715	.027	4.1	1.25	1:10	.758	.041	.706	.074	4.2	0.48	1:10
	GS	.749	.014	.696	.033	3.9	1.82	2:27	.789	.064	.723	.099	39.2	4.92	2:27
	LFS	.702	.038	.647	.075	1.7	0.82	0:31	.731	.017	.693	.042	2.3	1.41	0:31
	RS	.687	.041	.619	.093	1.3	0.03	2:03	.773	.025	.715	.058	23.1	15.82	2:03
	SSFS	6.98	.033	6.36	.061	1.1	0.05	0:28	.724	.043	.698	.093	3.5	1.54	0:28
	GPFR	.771	.037	.735	.046	18.3	4.83	316:01	7.92	.018	.731	.043	7.9	2.11	316:01
	GPFR-BC	-	-	.725	.083	3.1	1.27	316:01	-	-	.767	.073	14.7	2.53	316:01
NB	BF	.756	.032	.706	.057	4.3	1.82	0:59	.735	.044	.687	.071	3.2	1.71	0:59
	GS	.713	.013	.686	.088	24.8	12.53	2:25	.762	.027	.694	.084	38.5	5.22	2:25
	LFS	.679	.041	.627	.073	1.2	0.43	0:27	.741	.011	.673	.039	5.2	3.75	0:27
	RS	.682	.028	.618	.070	1.1	0.09	1:58	.694	.040	.658	.068	1.8	1.03	1:58
	SSFS	.673	.022	.621	.084	1.5	0.63	0:25	.705	.024	.670	.077	2,7	1.08	0:25
	GPFR	.758	.030	.716	.098	4.2	2.01	297:31	.738	.028	.697	.098	16.4	5.82	297:31
	GPFR-BC	-	-	.716	.120	99.2	5.32	297:31	-	-	.709	.048	21.1	5.22	297:31
3-NN	BF	.765	.013	.713	.093	5.3	1.23	1:19	.727	.026	.669	.087	2.8	1.18	1:19
	GS	.728	.034	.696	.085	6.9	3.42	2:45	.731	.025	.716	.046	40.3	8.34	2:45
	LFS	.673	.027	.608	.088	1.3	0.83	0:55	.694	.034	.671	.067	1.7	0.65	0:55
	RS	.682	.048	.611	.067	2.4	1.32	2:52	.691	.016	.640	.049	5.7	2.59	2:52
	SSFS	.677	.046	.601	.072	1.1	0.76	0:48	.689	.015	.635	.045	3.6	1.72	0:48
	GPFR	.796	.042	.745	.065	11.3	5.46	302:01	.792	.043	.728	.085	28.1	8.44	302:01
	GPFR-BC	-	-	.725	.072	2.0	0.10	302:01	-	-	.748	.062	41.6	7.28	302:01

Table 7.4: Comparison of FS wrappers and GPFR on Exostoses3 and Exostoses4 datasets.

results of comparison on data sets Exostoses3 and Exostoses4.

On both datasets, the GPFR methods outperformed the rest wrapper methods. The GPFR was better on Exostoses3, while the GPFR-BC on Exostoses4. From the wrapper classifier point of view, there is no significantly better classifier among the three used.

The last comparison on Exostoses data sets is shown in Table 7.5, whre results on Exostoses5 and Exostoses6 are shown. Here, again as in case of Exostoses3 and Exostoses4 the GPFR methods performed the best. In several cases, the GPFR-BC showed better results than the GPFR method, but majority was at the GPFR method side. If we analyse the three classifiers, non of them provided better wrapper for FS metod.

Now we can also analyse a time of run for particular method on particular classifier. The differences between classifiers are not high. The values of time needed for paticular method on particular classifier is averaged from 100 runs on all 6 exostoses datasets, because of the common features in datasets (except the class features). Here is the situation with performance opposite to the classification accuracy, because the GPFR method needs more than 300 hours to get such a good results in comparison to the rest of fs wrapper methods. This is the biggest disadvantage of the method. The GPFR-BC method need about the

same time, because it analyses the results of the GPFR method.

		Exostoses5 data							Exostoses6 data						
		Crit. Value		Classif. Rate		Subset size		Time	Crit. Value		Classif. Rate		Subset size		Time
Wrap.	FS Meth.	Mean	S.Dv.	Mean	S.Dv.	Mean	S.Dv.	(h:m)	Mean	S.Dv.	Mean	S.Dv.	Mean	S.Dv.	(h:m)
J48	BF	.773	.022	.711	.044	6.2	3.21	1:10	.843	.053	.767	.088	7.2	2.30	1:10
	GS	.847	.029	.758	.073	34.1	5.82	2:27	.802	.036	.758	.046	36.2	4.25	2:27
	LFS	.792	.031	.720	.056	8.4	3.26	0:31	.765	.026	.710	.047	4.5	1.72	0:31
	RS	.818	.038	.743	.080	9.7	4.28	2:03	.847	.029	.788	.083	25.2	5.25	2:03
	SSFS	.784	.019	.732	.054	8.1	1.39	0:28	.762	.059	.714	.072	2.5	1.04	0:28
	GPFR	.842	.034	.773	.053	13.2	2.42	316:01	.882	.042	.814	.099	14.3	3.85	316:01
	GPFR-BC	-	-	.762	.042	16.4	3.47	316:01	-	-	.808	.053	17.8	4.01	316:01
NB	BF	.829	.056	.718	.082	8.1	3.24	0:59	.871	.032	.780	.065	3.6	2.53	0:59
	GS	.835	.062	.741	.085	28.4	5.21	2:25	.825	.031	.753	.074	39.5	5.82	2:25
	LFS	.793	.020	.714	.052	2.6	1.19	0:27	.794	.034	.717	.069	4.7	2.15	0:27
	RS	.802	.040	.723	.064	3.4	1.32	1:58	.831	.038	.779	.087	12.9	5.26	1:58
	SSFS	.836	.016	.731	.033	14.3	3.73	0:25	.786	.019	.723	.038	14.3	4.72	0:25
	GPFR	.857	.026	.810	.075	12.5	4.21	297:31	.863	.045	.791	.096	8.3	2.18	297:31
	GPFR-BC	-	-	.795	.083	10.3	2.56	297:31	-	-	.783	.055	15.4	3.28	297:31
3-NN	BF	.742	.018	.682	.018	7.2	3.52	1:19	.852	.037	.779	.064	6.7	3.04	1:19
	GS	.785	.045	.723	.086	48.5	9.33	2:45	.795	.029	.742	.075	42.7	10.6	2:45
	LFS	.798	.023	.735	.096	4.6	2.52	0:55	.798	.013	.731	.042	4.9	2.34	0:55
	RS	.774	.025	.719	.071	5.2	3.71	2:52	.813	.048	.762	.093	27.4	6.55	2:52
	SSFS	.792	.029	.746	.073	23.5	7.69	0:48	.810	.051	.747	.065	5.2	2.37	0:48
	GPFR	.840	.032	.779	.046	14.5	4.72	302:01	.847	.017	.788	.081	9.7	2.17	302:01
	GPFR-BC	-	-	.769	.057	12.3	2.71	302:01	-	-	.789	.121	23.1	4.22	302:01

Table 7.5: Comparison of FS wrappers and GPFR on Exostoses5 and Exostoses6 datasets.

7.5 Comparison on Madelon data set

In this section, the GPFR method is compared to all kinds of FS methods as stated in the introduction of the state of the art. The wrapper methods are the same as in the previous section, the embedded method is the first contribution of this thesis, the FeRaNGA-7 method, and finally the following filter methods are also used: InfoGain, SVM and ReliefF.

The madelone data set was used as the biggest dataset in this thesis, because of time peformance, which is not a strong side of out ML based approach. The results are shown in table 7.6. First impotant result in this experiment is the lower classifier performance of NB, where in comparison to the remaining classifiers the best CA is about the same as the worst wrapper CA on J48 and 3-NN.

The best CAs among wrappers are computed by 3-NN classifier, therefore this classifier was also used for computation of CAs for the three filter methods as shown at bottom of the table. For the 3-NN wrapper, the GPFR method outperformed all the methods

		Madelone data						
		Crit. Value		Classif. Rate		Subset size		Time
Wrap.	FS Meth.	Mean	S.Dv.	Mean	S.Dv.	Mean	S.Dv.	(h)
J48	BF	.763	.158	.719	.177	21.7	4.52	1.8
	GS	.738	.042	.687	.058	175.4	18.43	0.6
	LFS	.725	.152	.645	.029	5.8	2.05	0,0
	RS	.697	.130	.734	.153	315.2	24.84	1.2
	SSFS	.829	.098	.777	.061	15.2	6.79	0.0
	GPFR	.849	.118	.812	.023	17.4	5.21	231.4
	GPFR-BC	-	-	.809	.148	19.3	3,42	231.4
NB	BF	.651	.062	.581	.090	22.1	4.8	1.6
	GS	.648	.093	.576	.075	185.3	10.4	0.6
	LFS	.672	.148	.607	.142	8.3	2.4	0.0
	RS	.649	.073	.610	.179	16.2	3.1	1.1
	SSFS	.673	.034	.605	.151	7.2	1.2	0.0
	GPFR	.659	.153	.607	.186	12.2	2.65	224.8
	GPFR-BC	-	-	.621	.191	5.2	0.31	224.8
3-NN	BF	.693	.071	.587	.039	13.7	4.82	1.8
	GS	.591	.020	.543	.127	28.1	6.90	0.8
	LFS	.657	.038	.602	.095	5.4	2.05	0.0
	RS	.689	.019	.741	.196	9.2	1.12	1.2
	SSFS	.673	.048	.603	.183	4.7	1.08	0.0
	GPFR	.935	.039	.890	.080	17.3	2.81	233.6
	GPFR-BC	-	-	.885	.197	16.4	3.05	233.6
		Madelone data						
Embed.	FeRaNGA-7	.838	.137	.781	.120	25.4	7.36	76.6
Filter	InfoGain	.753	.021	.727	.072	8.2	3.16	0.0
	SVM	.782	.116	.734	.156	310.2	83.58	0.4
	ReliefF	.829	.136	.784	.019	15.4	4.20	0.0

Table 7.6: Comparison of all three types of FS methods on Madelon data set.

in comparison (except the GPFR-BC with the second best CA). The embedded method FeRaNGA-7 have comparable results to filters provides better results that almost all the wrappers. The time comparison on madelon data is again the worst for the GPFR methods.

The FeRaNGA method is the second worst with circa 1/3 of the GPFR's time.

7.6 Time Performance Evaluation

Evaluation of the fitness is the most time consuming part of the GPFR and GPFS algorithms. The evaluation of the fitness can be divided into two components. First component represents the evaluation of the evolved scoring function for each feature, and the second component is the evaluation of the feature ranking result provided by the scoring function.

The first part of the fitness evaluation is for the Hyprecube data set evaluated for each scoring function 50 times (because of 50 features in the data set). The median time for

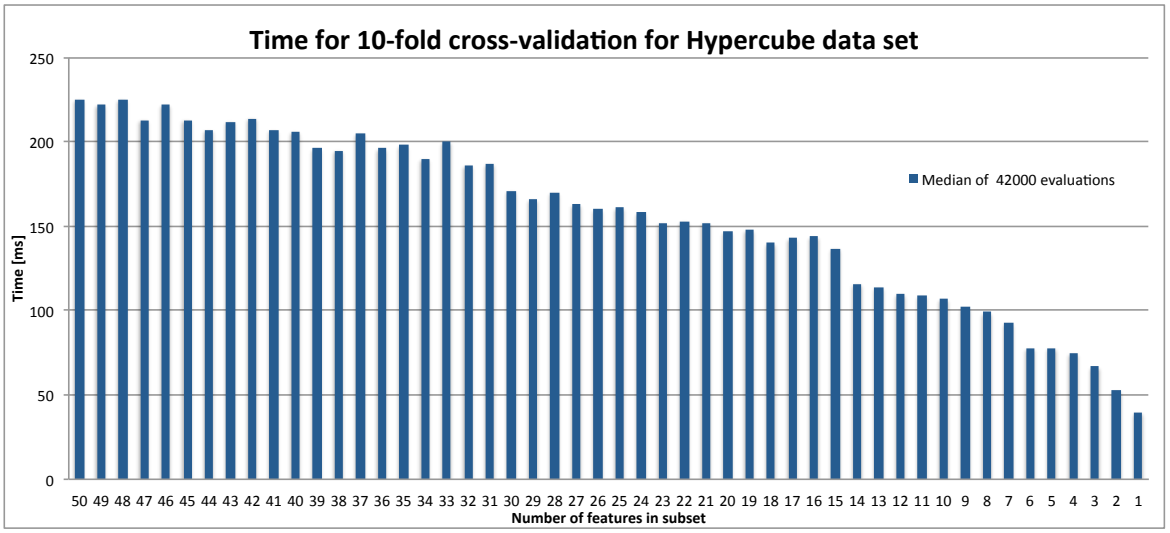

Figure 7.1: This Figure shows the time needed for evaluation of the second part of the GPFR fitness on Hypercube data set. There were 42000 of evaluations for each size of subset from 50 (all features) to 1 (ontly the best one feature). The median value for the smallest subset is about 5 times smaller than for the full subset of features.

these 50 evaluation computed from 100 individuals is 77188,42 ns. For constants or "small" functions the time is 8 ns, but for the most complex functions the time is about 200 ms.

The second part of the fitness evaluation is presented in Fig. 7.1. For the best one feature in the subset the algorithm need about 40 ms to cross-validate the model. For the full set of features (50) it is about 250 ms.

The second part of the fitness is therefore more time consuming than the first part. To get the time needed for evaluation of the whole fitness for one individual we have to add the first part of the fitness to the sum of the times needed for all subsets of the data set.

We have performed the same evaluation also for the Ionosphere dataset, QSAR Biodegradation dataset and for the oral exostoses data set and the average time for evaluation of one attribute on the same number of instances (to have comparable results) was the same in average with this example.

Chapter 8

Summary, Conclusions and Future Work

8.1 Summary and Conclusions

This thesis provides the general overview of artificial neural networks and genetic programming (chapter 2) needed to create the main contributions of this thesis - two new methods for feature ranking and feature selection called FeRaNGA (Feature Ranking derived from Niching Genetic Algorithm; chapter 3) and GPFR (Genetic Programming based Feature Ranking; chapter 4). Both methods are able to perform also feature selection and to achieve results comparable to and in many cases better that the actual methods of the-state-of-the-art.

The FeRaNGA method fulfilled one of the main goals of this thesis and proved, that it is possible to use the GAME artificial neural network for automatic reduction of input features number and for ranking of all selected features according to their importance. Moreover, this method does not need any additional computation, all information for ranking and selection are extracted from the process of the GAME data mining model evolution. Further advantage is that it can be applied to regression problems as well as to classification problems. FeRaNGA simplifies the whole process of data mining for researches from different areas of science without expert knowledge of data mining (especially of methods for feature ranking and feature selection). The only thing they have to do is to create the data mining model and get all the results at once: subset of the most important fea-

tures, ranks for all selected features and results of the classification or regression model with already included FeRaNGA results of the feature ranking and feature selection. The FeRaNGA method was successfully applied in the problem of dental age prediction and tooth importance estimation and the results were presented in the impacted journal [A.1].

The GPFR method fulfilled the second main goal of this thesis and demonstrated that it is possible to use genetic programming to evolve a data-set-specific mathematical expression representing feature scoring function for feature importance determination and feature selection. Success of this approach allows reducing the amount of specific data mining knowledge in modeling process. GPFR provides a complex data-specific solution of data mining problem. This solution covers a data mining model as well as the results of feature selection and feature ranking. Further advantage of this genetic programming-based approach is the evolved mathematical expression that describes the importance of input attributes. The GPFR method is able to solve not only the classification task but could be also used for the regression tasks. This regression part was not tested yet and is planed for future work. We have successfully applied this method to a real-world problem of oral exostoses modeling and submitted a publication to the impacted journal [A.2].

The ML approach proposed in this thesis was also used another application: "Disregarding population-specificity: an influence on the sex assessment methods from the tibia" submitted to impact journal [A.3].

The Borda count technique for combining of feature ranking and feature selection results was examined and is presented as a suitable technique in real-world application of the GPFR method to the problem of oral exostoses modeling.

The goals stated in chapter 1.2 were all fulfilled. After an extensive study of the ANN and GP areas, two new approaches for feature ranking and feature selection (FeRaNGA and GPFS) were proposed, implemented and successfully tested on synthetic and real-world data sets. Furthermore, they were successfully tested in real world applications for dental age estimation and teeth importance determination and for oral exostoses modeling and identifying of the relevant features influencing their presence.

8.2 The Contribution

- Two new feature ranking and feature selection methods (FeRaNGA and GPFR) have been proposed.
- FeRaNGA and GPFR were successfully tested and compared with other state-of-the-art methods for feature ranking and feature selection on synthetic and real-world data sets.
- The Borda count was examined and used as a suitable technique for combining of feature selection and feature ranking results.
- FeRaNGA was successfully applied to the real-world problem of dental age estimation and teeth importances determination.
- GPFR was successfully applied to the real-world problem of oral exostoses modeling and the most important features determination.
- The presented approach reduces data dimensionality and simplifies the use of data mining by researchers without expert knowledge of this domain.

Chapter 9

Suggestions for future research

Although a large number of experiments have been performed, and a lot of various modifications of the algorithms and settings have been tested, there are still opportunities for the future research. In following sections, there are some suggestions for further research in the area of FeRaNGA, GPFR and Borda count.

9.1 Suggestions in the area of FeRaNGA method

- Examination of feature rankings and their importances from higher layers of the network in general.
- To consider the penalization of the features in higher layers in contrast to the ranks from the first layer.

9.2 Suggestions in a field of genetic programming-based approach

The GPFS method is the youngest part of this thesis and although this approach achieved very good results during first experiments and was successfully applied in real world problem, there is a lot of possibilities how to extend or modify the current method. Following items are examples and points of what could be more deeply investigated, improved or modified:

- Modification of the GPFR fitness reflecting positive and negative classification accuracy trend changes and penalization of big falls in CA
- Experiments with lower number of features used in fitness evaluation process.
- Implementation of a policy for the fitness to deal with a situation when two or more features are scored with the same value while having a high inter-correlation among them (feature redundancy).
- Substitution of the classifier in fitness computation with a regression model and investigation of what performance offers GPFR method in a field of regression problems.

Bibliography

[1] F. H. Salvador García, Julián Luengo, *Data Preprocessing in Data Mining*, vol. 72. Intelligent Systems Reference Library, 2015.

[2] J. Biesiada, W. Duch, A. Kachel, K. Maczka, and S. Palucha, "Feature ranking methods based on information entropy with parzen windows," pp. 109–119, 2005.

[3] H. Liu, H. Motoda, R. Setiono, and Z. Zhao, "Feature selection an ever evolving frontier in data mining," vol. 10, pp. 4–13, 2010.

[4] I. Guyon and A. Elisseeff, "An introduction to feature extraction," in *Feature Extraction* (G. Isabelle, N. Masoud, G. Steve, and Z. LotfiA., eds.), vol. 207 of *Studies in Fuzziness and Soft Computing*, pp. 1–25, Springer Berlin Heidelberg, 2006.

[5] J. Liang, S. Yang, and A. Winstanley, "Invariant optimal feature selection: A distance discriminant and feature ranking based solution," *Pattern Recognition*, vol. 41, pp. 1429–1439, 2008.

[6] Sayes, Inza, and Larranaga, "A review of feature selection techniques in bioinformatics," *Bioinformatics*, vol. 23, pp. 2507–2517, 2007.

[7] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial Intelligence journal, special issue on relevance*, vol. 97, pp. 273–324, 1997.

[8] P. A. Devijver and J. Kittler, "Pattern recognition: A statistical approach," *Prentice Hall*, 1982.

[9] R. Price, "Essay towards solving a problem in the doctrine of chances," *Philosophical Transactions of the Royal Society of London*, 1764.

[10] V. F. J. Kittler and L. Pau, *Pattern Recognition Theory and Applications*. Proceedings of the NATO Advanced Study Institute held at St. Annes College in Oxford, 1981.

[11] A. R. Webb, *Statistical Pattern Recognition*. John Wiley and Sons, 2006.

[12] M. AJ, "Selection of subsets of regression variables," *Journal of the Royal Statistical Society, Series A-G*, pp. 389–425, 1984.

[13] P. Kordík, *Fully Automated Knowledge Extraction using Group of Adaptive Models Evolution*. PhD thesis, Czech Technical University in Prague, FEE, Dep. of Comp. Sci. and Computers, FEE, CTU Prague, Czech Republic, September 2006.

[14] N. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proceedings of the First International Conference on Genetic Algorithms*, pp. 183–187, 1985.

[15] D. Dickmanns, J. Schmidhuber, and A. Winklhofer, "Der genetische algorithmus: Eine implementierung in prolog," *Fortgeschrittenenpraktikum, Institut f ur Informatik, Lehrstuhl Prof. Radig, Technische Universit at M unchen*, 1987.

[16] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[17] U. Fayyad, G. Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, pp. 37–54, 1996.

[18] I. Witten and E. Frank, *Data Mining - Practical Machine Learning Tools and Techniques, Second Edition*. Elsevier, 2005.

[19] S. Kotsiantis, "Supervised machine learning: A review of classification techniques," *Informatica*, vol. 31, pp. 249–268, 2007.

[20] R. S. M. T. Mitchell, J. G. Carbonell, "Book review: Machine learning: A guide to current research by," *SIGART Bull.*, pp. 22–, Jan. 1988. Reviewer-Price, Keith.

[21] "Book review: Machine learning, neural and statistical classification edited by d. michie, d.j. spiegelhalter and c.c. taylor (ellis horwood limited, 1994)," *SIGART Bull.*, vol. 7, pp. 16–17, Jan. 1996.

[22] P. P. F.J. Ferri and M. Hafter, "Comparative study of techniques for large-scale feature selection," *Pattern Recognition in Practice, IV: Multiple Paradigms, Comparative Studies and Hybrid*, 2001.

[23] H. Liu and H. Motoda, *Computational Methods of Feature Selection*. Chapman & Hall/CRC, 2008.

[24] R. Kohavi and G. John, "Wrappers for feature subset selection," *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[25] A. L. Blum and P. Langley, "Selection of relevant features and examples in machine learning," *ARTIFICIAL INTELLIGENCE*, vol. 97, pp. 245–271, 1997.

[26] S. H. Huang, "Supervised feature selection: A tutorial," *Afrtifical Inteligence Research*, vol. 4, pp. 22–37, 2015.

[27] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[28] H. Stoppiglia, G. Dreyfus, R. Dubois, and Y. Oussar, "Ranking a random feature for variable and feature selection," *J. Mach. Learn. Res.*, vol. 3, pp. 1399–1414, Mar. 2003.

[29] P. Pudil, J. Novovičová, and J. Kittler, "Floating search methods in feature selection," *Pattern Recogn. Lett.*, vol. 15, pp. 1119–1125, Nov. 1994.

[30] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *Knowledge and Data Engineering*, vol. 17, pp. 419–502, 2005.

[31] F. L. Peng, H. and D. C., "Feature selection based on mi criteria of max-dependency, max-relevance, and min-redundancy," *Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1226–1238, 2005.

[32] R. T. Collins, Y. Liu, and M. Leordeanu, "Online selection of discriminative tracking features," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, pp. 1631–1643, Oct. 2005.

[33] R. R. Hocking, "A biometrics invited paper. the analysis and selection of variables in linear regression," *Biometrics, International Biometric Society*, pp. 1–49, 1976.

[34] A. Kalousis, J. Prados, and M. Hilario, "Stability of feature selection algorithms: A study on high-dimensional spaces," *Knowl. Inf. Syst.*, vol. 12, pp. 95–116, May 2007.

[35] L. I. Kuncheva, "A stability index for feature selection," in *Proceedings of the 25th Conference on Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications*, AIAP'07, (Anaheim, CA, USA), pp. 390–395, ACTA Press, 2007.

[36]

[37] Y. Saeys, T. Abeel, and Y. Peer, "Robust feature selection using ensemble feature selection techniques," ECML PKDD 08, pp. 313–325, Springer-Verlag, 2008.

[38] J. Reunanen, "Overfitting in making comparisons between variable selection methods," *J. Mach. Learn. Res.*, vol. 3, pp. 1371–1382, 2003.

[39] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[40] M. Hall, "Feature subset selection: A correlation based filter approach," *International Conf. on Machine Learning*, vol. 17, 1997.

[41] W. Punch, E. Goodman, M. Pei, L. Chia-Shun, P. Hovland, and R. Enbody, "Further research on feature selection and classification using genetic algorithms," 1993.

[42] G. Castellano and A. M. Fanelli, "Variable selection using neural-network models," *Neurocomputing*, vol. 31, pp. 1–13, 1999.

[43] R. Harper and A. Blair, "Dynamically defined functions in grammatical evolution," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pp. 2638–2645, 2006.

[44] J. H. Hong and S. Cho, "Cancer prediction using diversity-based ensemble genetic programming," in *Modeling Decisions for Artificial Intelligence* (V. Torra, Y. Narukawa, and S. Miyamoto, eds.), vol. 3558 of *Lecture Notes in Computer Science*, pp. 294–304, Springer Berlin Heidelberg, 2005.

[45] D. Muni, N. Pal, and J. Das, "A novel approach to design classifiers using genetic programming," *Evolutionary Computation, IEEE Transactions on*, vol. 8, pp. 183–196, April 2004.

[46] K. Neshatian, M. Zhang, and P. Andreae, "Genetic programming for feature ranking in classification problems," in *Simulated Evolution and Learning*, vol. 5361 of *Lecture Notes in Computer Science*, pp. 544–554, Springer Berlin Heidelberg, 2008.

[47] S. Ahmed, M. Zhang, and L. Peng, "Feature selection and classification of high dimensional mass spectrometry data: A genetic programming approach," in *Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics* (L. Vanneschi, W. Bush, and M. Giacobini, eds.), vol. 7833 of *Lecture Notes in Computer Science*, pp. 43–55, Springer Berlin Heidelberg, 2013.

[48] H. Liu and H. Motoda, *Computational Methods of Feature Selection*. Chapman & Hall/CRC, 2008.

[49] J. A. Muller and F. Lemke, *Self-Organising Data Mining*. Berlin, 2000. ISBN 3-89811-861-4.

[50] S. Piramuthu, "Evaluating feature selection methods for learning in data mining applications," *European Journal of Operational Research*, vol. 156, pp. 483–494, 2004.

[51] W. Siedlecki and J. Sklansky, "On automatic feature selection," *International Journal of Pattern Recognition*, vol. 2, pp. 197–220, 1988.

[52] M. H. J. Grim, P. Somol and P. Pudil, "Color texture segmentation by decomposition of gaussian mixture model," *Lecture Notes in Computer Science*, vol. 19, pp. 287–296, 2006.

[53] M. Hall, "Correlation-based feature subset selection for machine learning," *PhD dissertation, Department of Computer Science, University of Waikato*, 1999.

[54] M. Robnink-Sikonja and I. Kononenko, "An adaptation of relief for attribute estimation in regression," 1994.

[55] R. Battiti, "Using mutual information for selecting features in supervised neural net learning," *IEEE TRANSACTIONS ON NEURAL NETWORKS*, vol. 5, NO. 4, 1994.

[56] M. Tesmer and P. Estevez, "Amifs: adaptive feature selection by using mutual information," in *Proceedings of the 2004 IEEE International Joint Conference on Neural Networks*, vol. 1, (Dept. of Electr. Eng., Chile Univ., Santiago, Chile), p. 308, July 2004.

[57] S. S. HAYKIN, *Neural networks and learning machines. 3rd ed.* Upper Saddle River: Pearson, 2009.

[58] T. Kohonen, "Self-organized formation of topologically correct feature maps," pp. 59–69, 1982.

[59] H. Madala and A. Ivakhnenko, *Inductive Learning Algorithm for Complex System Modelling.* CRC Press, 1994. Boca Raton.

[60] P. Kordík, *Fully Automated Knowledge Extraction using Group of Adaptive Models Evolution.* PhD thesis, Czech Technical University in Prague, FEE, Dep. of Comp. Sci. and Computers, FEE, CTU Prague, Czech Republic, September 2006.

[61] E. S. A. Ivakhnenko and G. Ivakhnenko, "Gmdh algorithm for optimal model choice by the external error criterion with the extension of definition by model bias and its applications to the committees and neural networks.," *Pattern Recognition and Image Analysis*, vol. 12(4):347353, 2002.

[62] J. A. Muller and F. Lemke, "Self-organising data mining," *Berlin 2000*, vol. ISBN 3-89811-861-4., 2000.

[63] K. P., *GAME - Hybrid Self-Organizing Modeling System based on GMDH*, vol. 1 of *Studies in Computational Intelligence*, ch. GAME - Hybrid Self-Organizing Modeling System based on GMDH, pp. 290–340. Czech Technical University in Prague, FEE, Dep. of Comp. Sci. and Computers: Springer-Verlag, Berlin, Heidelberg, 2009.

[64] A. Eiben and J. Smith, *Introduction to Evolutionary Computing.* Springer Verlag, 2003.

[65] L. Fogel, A. Owens, and M. Walsh, "Artificial intelligence through simulated evolution," *Wiley, Chichester, UK*, 1996.

[66] J. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM J. of Computing, 2*, pp. 88–105, 1973.

[67] K. D. Jong, M. Potter, and W. Spears, "Using problem generators to explore the effects of epistasis.," *Back 23*, pp. 338–345.

[68] I. Rechenberg., "Evolutionstrategie: Optimierung technisher systeme nach prinzipien des biologischen evolution," *Fromman-Hozlboog Verlag, Stuttgart*, 1973.

[69] "Evolving objects library - online tutorial," 2006.

[70] R. Poli, W. Langdon, N. McPhee, and J. Koza, *A field guide to genetic programming*. February 2015.

[71] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.

[72] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford, UK: Oxford University Press, 1996.

[73] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1998.

[74] W. Banzhaf, ed., *Foundations of Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1st ed., 1999.

[75] S. W. Mahfoud, "Niching methods for genetic algorithms," Tech. Rep. 95001, Illinois Genetic Algorithms Laboratory (IlliGaL), University of Ilinios at Urbana-Champaign, May 1995.

[76] S. W. Mahfoud, "A comparison of parallel and sequential niching methods," in *Sixth International Conference on Genetic Algorithms*, pp. 136–143, 1995.

[77] O. J. Mengshoel and D. E. Goldberg, "The crowding approach to niching in genetic algorithms," *Evol. Comput.*, vol. 16, pp. 315–354, Sept. 2008.

[78] G. Brown, *Diversity in Neural Network Ensembles*. PhD thesis, The University of Birmingham, School of Computer Science, Birmingham B15 2TT, United Kingdom, January 2004.

[79] F. Otero, M. Silva, A. Freitas, and J. C. Nievola, "Genetic programming for attribute construction in data mining," in *Genetic Programming*, pp. 384–393, Springer, 2003.

[80] M. Rizki, M. Zmuda, and L. Tamburino, "Evolving pattern recognition systems," *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 6, pp. 594–609, 2002.

[81] H. Gray, R. Maxwell, I. Martínez-Pérez, C. Arus, and S. Cerdan, "Genetic programming for classification and feature selection: analysis of 1h nuclear magnetic resonance

spectra from human brain tumour biopsies," *NMR in Biomedicine*, vol. 11, no. 4-5, pp. 217–224, 1998.

[82] M. Smith and L. Bull, "Using genetic programming for feature creation with a genetic algorithm feature selector," in *Parallel Problem Solving from Nature-PPSN VIII*, pp. 1163–1171, Springer, 2004.

[83] J. Lin, H. Ke, B. Chien, and W. Yang, "Classifier design with feature selection and feature extraction using layered genetic programming," *Expert Syst Appl.,2008,34.*, pp. 1384–1393, 2008.

[84] K. Neshatian, M. Zhang, and P. Andreae, "Genetic programming for feature ranking in classification problems," in *Simulated Evolution and Learning*, pp. 544–554, Springer, 2008.

[85] P. L. Ahmed S, Zhang M, "Evolutionary computation, machine learning and data mining in bioinformatics," *Lecture Notes in Computer Science*, vol. 7833, pp. 44–44, 2013.

[86] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[87] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA, USA: MIT Press, 1994.

[88] N. Chawla, "Data mining for imbalanced datasets: An overview," in *Data mining and knowledge discovery handbook*, pp. 853–867, Springer, 2005.

[89] M. V. Erp and L. Schomaker, "Variants of the borda count method for combining ranked classifier hypotheses," in *In the seventh international workshop on frontires in handwriting recognition. Amsterdam learning methodology ispired by human's intelligence*, Citeseer, 2000.

[90] W. Yan, "Fusion in multi-criterion feature ranking," in *Information Fusion, 2007 10th International Conference on*, pp. 1–6, IEEE, 2007.

[91] P. Baldi, S. Brunak, Y. Chauvin, C. A. F. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview," *Bioinformatics*, vol. 16, no. 5, pp. 412–424, 2000.

[92] K. Polat and S. Guene, "A new feature selection method on classification of medical datasets: Kernel f-score feature selection," *Expert Systems with Applications*, vol. 36, no. 7, pp. 10367 – 10373, 2009.

[93] D. J. Hand, "Measuring classifier performance: A coherent alternative to the area under the roc curve," *Machine Learning*, vol. 77, pp. 103 – 123, 2009.

[94] "Uci machine learning repository." available at http://www.ics.uci.edu/ mlearn/ML-Summary.html, September 2006.

[95] C. Moorrees, E. Fanning, and E. Hunt, "Age variation of formation stages for ten permanent teeth.," *Dent. Res. 42*, p. 14901502, 1963.

[96] J. Velemínská, A. Pilný, M. Čepek, M. Kot'ová, and R. Kubelková, "Dental age estimation and different predictive ability of various tooth types in the czech population: data mining methods," *Anthropologischer Anzeiger*, vol. 70, no. 3, pp. 331–345, 2013.

[97] P. Somol and J. Novovicova, "Evaluating stability and comparing output of feature selectors that optimize feature subset cardinality," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 1921–1939, Nov. 2010.

Chapter 10

Publications

10.1 Refereed publications of the author

Bibliography

10.1.1 Impact publications

[A.1] J. Velemínská, A. Pilný, M. Čepek, M. Koťová, R. Kubelková. Dental age estimation and different predictive ability of various tooth types in the Czech population: data mining methods *In: Anthropologischer Anzeiger* , 1/2013, 3-70, p. 331-345. ISSN 0003-5548 Main author: A. Pilný. Proportions 25%, 25%, 25%, 5%, 20%.

[A.2] A. Pilný, Z. Buk, A. Léonard, P. Bayle, P. Kordík, M. Šnorek., J. Brůžek. Genetic Programming-Based Feature Selction applied in Oral Exostoses Modeling — *awaiting acceptance notification*, 2015. (eqal)

[A.3] M. Koťerová,J. Velemínská, J. Dupej, H. Brzobohatá, A. Pilný, J. Brůžek. Disregarding population-specificity: an influence on the sex assessment methods from the tibia. — *awaiting acceptance notification*, 2015. (eqal)

10.1.2 Refereed publications in proceedings

[A.4] A. Pilný, P. Kordík P., M. Šnorek. Feature Ranking Derived from Data Mining Process. *Proceedings of the ICANN 2008*, Heidelberg: Springer, 2008, vol. 1, p. 889-898. ISBN 978-3-540-87558-1. (equal)

[A.5] A. Pilný, Kordík P., Šnorek M. GAME Model Utilization for Feature Ranking and Feature Selection *Proceedings of the 7th EUROSIM Congress on Modelling and Simulation*, Vol. 2: Full Papers, 2010, p. 143-148. ISBN 978-80-01-04589-3. (equal)

[A.6] Cepek M., Pilný A., Kubelková R., Velemínská J., Šnorek M., Modelling of Age from the Teeth Development *Proceedings of the 7th EUROSIM Congress on Modelling and Simulation, Vol. 2: Full Papers, 2010* p. 118 - 125. ISBN 978-80-01-04589-3. (eqal)

[A.7] A. Pilný, W. Oertel, P. Kordík, M. Šnorek. Correlation Based Feature Ranking in Combination with Embedded Feature Selection. International Workshop on Inductive Modeling IWIM 2009, Krynica, Poland. (equal)

[A.8] A. Pilný, Kordík P., Šnorek M. Behaviour of FeRaNGA method for Feature Ranking during learning process using Inductive Modelling *Proceedings of the 2nd International Conference on Inductive Modelling ICIM 2008. Kiev: Ukr.*, INTEI, 2008, p. 222-226. ISBN 978-966-02-4889-2., (equal)

[A.9] A. Pilný, P. Kordík, M. Šnorek, R. Kubelková Aplication of Feature Selection in Age Prediction. International Conference on Inductive Modelling - ICIM 2010, Yevpatoria, Ukraine. (equal)

10.1.3 Other publications

[A.10] A. Pilný, W. Oertel, P. Kordík, M. Šnorek. New Methods for Feature Ranking. CTU Workshop 2010, Prague, Czech Republic, p. 102-103. ISBN 978-80-01-04513-8.(equal)

10.1.4 Citations

- Paper [A.4] has been cited in impacted publication:

  1. T. Řehořek, P. Kordík. A Soft Computing Approach to Knowledge Flow Synthesis and Optimization. *Computer Journal*, 33:29–35,2013, ISBN 978-3-642-32921-0.

  Paper [A.4] has been self-cited in impact publication:

  1. J. Velemínská, A. Pilný, M. Čepek, M. Kotová, R. Kubelková. Dental age estimation and different predictive ability of various tooth types in the Czech population: data mining methods *In: Anthropologischer Anzeiger* , 1/2013, 3-70, p. 331-345. ISSN 0003-5548

10.2 Other refereed publications of the author

Bibliography

10.2.1 Authorised software

[A.11] P. Kordík P., M. Čepek, J. Drchal, O. Kovářik, A. Pilný. FAKE-GAME - authorized data mining open source software : 2009. Online: fakegame.sourceforge.net. (equal)

10.2.2 Refereed publications in proceedings

[A.12] D. Novák, A. Pilný, Kordík P., Š. Holiga, P. Posik, et al. Analysis of Vestibular-Ocular Reflex by Evolutionary Framework *International Conference on Artificial Neural Networks (ICANN) 2008, 18th International Conference Proceedings*, Heidelberg: Springer, 2008, vol. 1, p. 452-461. ISBN 978-3-540-87535-2.(equal)

[A.13] A. Pilný, P. Kordík Reconstruction of Eye Movements Signal using Inductive Model Detecting Saccades. *In proceedings of IWIM 2007, International Workshop on Inductive Modelling, Prague, Czech Rep.*, 2007.(equal)

Appendix A

Results of FeRaNGA-n and GPFR method

Table A.1: Results of FeRaNGA-n algorithm on Hypercube data set for different number of models with NGA configuration 150 individuals in population and 150 epochs. Results are shown for all layers generated by the GAME algorithm. Rows denoted as overall represent ranks computed from all the generated layers of the model. All the ranks in first layers are correct with between 8 and 9 features selected.

	Layer	Ideal ranking: 1 to 10, 21 to 50, 11 to 21												
FeRaNGA-1	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	48	35	39	40	45
	Overall	1	2	3	4	5	6	7	8	48	35	39	40	45
FeRaNGA-2	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	40	44	45	35	42	30
	Overall	1	2	3	4	5	6	7	8	40	44	45	35	42
FeRaNGA-3	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	40	44	45	35	42	30
	Overall	1	2	3	4	5	6	7	8	40	44	45	35	42
FeRaNGA-4	0	1	2	3	4	5	6	7	8	9				
	1	1	2	3	4	5	6	7	8	49	12	22	24	25
	Overall	1	2	3	4	5	6	7	8	49	9	12	22	24
FeRaNGA-5	0	1	2	3	4	5	6	7	8	9				
	1	1	2	3	4	5	6	7	49	8	12	22	26	30
	Overall	1	2	3	4	5	6	7	8	9	49	12	22	26
FeRaNGA-6	0	1	2	3	4	5	6	7	8	9				
	1	1	2	3	4	5	6	7	8	26	30	49	12	13
	2	1	39	2	3	4	13	6	48	5	50	26	24	32
	Overall	1	2	3	4	5	6	7	8	26		22	24	30
FeRaNGA-7	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	26	30	49	12	13
	2	1	39	2	3	4	13	6	48	5	50	26	24	32
	Overall	1	2	3	4	5	6	7	8	26	13	22	24	30

Table A.2: Results of FeRaNGA-n algorithm on Hypercube data set for different number of models with NGA configuration 150 individuals in population and 150 epochs. Number of models used was from 1 up to 14 (FeRaNGA-14). Results are shown for all layers generated by the GAME algorithm. Rows denoted as overall represent ranks computed from all the generated layers of the model. All the ranks in first layers are correct with number of selected features between 6 and 9.

	Layer	Ideal ranking: 1 to 10, 21 to 50, 11 to 21												
FeRaNGA-1	0	1	2	3	4	5	7	6	8	9				
	1	1	2	3	4	5	7	6	9	8	42	44	50	18
	0verall	1	2	3	4	5	7	6	9	8	42	44	50	18
FeRaNGA-2	0	1	2	3	4	5	7	6	8	9	37			
	1	1	2	3	4	5	7	6	9	8	42	44	50	18
	0verall	1	2	3	4	5	7	6	9	8	37	42	44	50
FeRaNGA-3	0	1	2	3	4	5	7	6						
	1	1	2	3	4	5	6	7	8	50	9	23	38	14
	0verall	1	2	3	4	5	7	6	8	50	9	23	38	14
FeRaNGA-4	0	1	2	3	4	5	7	6	8					
	1	1	2	3	4	5	6	7	8	50	9	23	38	14
	0verall	1	2	3	4	5	7	6	8	50	9	23	38	14
FeRaNGA-5	0	1	2	3	4	5	7	6						
	1	1	2	3	4	5	6	7	8	44	50	13	14	15
	0verall	1	2	3	4	5	7	6	8	44	50	13	14	15
FeRaNGA-6	0	1	2	3	4	5	7	6	8					
	1	1	2	3	4	5	6	7	8	44	50	13	14	15
	0verall	1	2	3	4	5	7	6	8	44	50	13	14	15
FeRaNGA-7	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	44	50	13	14	15
	0verall	1	2	3	4	5	6	7	8	44	50	13	14	15
FeRaNGA-8	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	44	50	13	14	15
	0verall	1	2	3	4	5	6	7	8	44	50	13	14	15
FeRaNGA-9	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	9	44	50	13	14
	2	1	2	3	4	5	44	20	9	8	26	41	42	45
	0verall	1	2	3	4	5	6	7	8	9	44	47	50	13
FeRaNGA-10	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	9	44	50	13	14
	2	1	2	3	4	5	44	20	9	8	26	41	42	45
	0verall	1	2	3	4	5	6	7	8	9	44	47	50	13
FeRaNGA-11	0	1	2	3	4	5	6	7						
	1	1	2	3	4	5	6	7	8	9	44	50	13	14
	2	1	2	3	4	5	44	20	9	8	26	41	42	45
	0verall	1	2	3	4	5	6	7	8	9	44	47	50	13
FeRaNGA-12	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	9	44	13	14	15
	2	1	2	3	4	5	44	20	9	8	26	41	42	45
	0verall	1	2	3	4	5	6	7	8	9	44	15	20	21
FeRaNGA-13	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	38	44	49	50	9
	2	1	2	4	3	5	6	8	7	26	49	44	18	20
	0verall	1	2	4	3	5	6	7	8	49	44	50	9	15
FeRaNGA-14	0	1	2	3	4	5	6	7	8					
	1	1	2	3	4	5	6	7	8	14	15	18	19	21
	2	1	2	4	3	5	6	8	7	26	49	44	18	20
	0verall	1	2	4	3	5	6	7	8	15	18	29	40	41

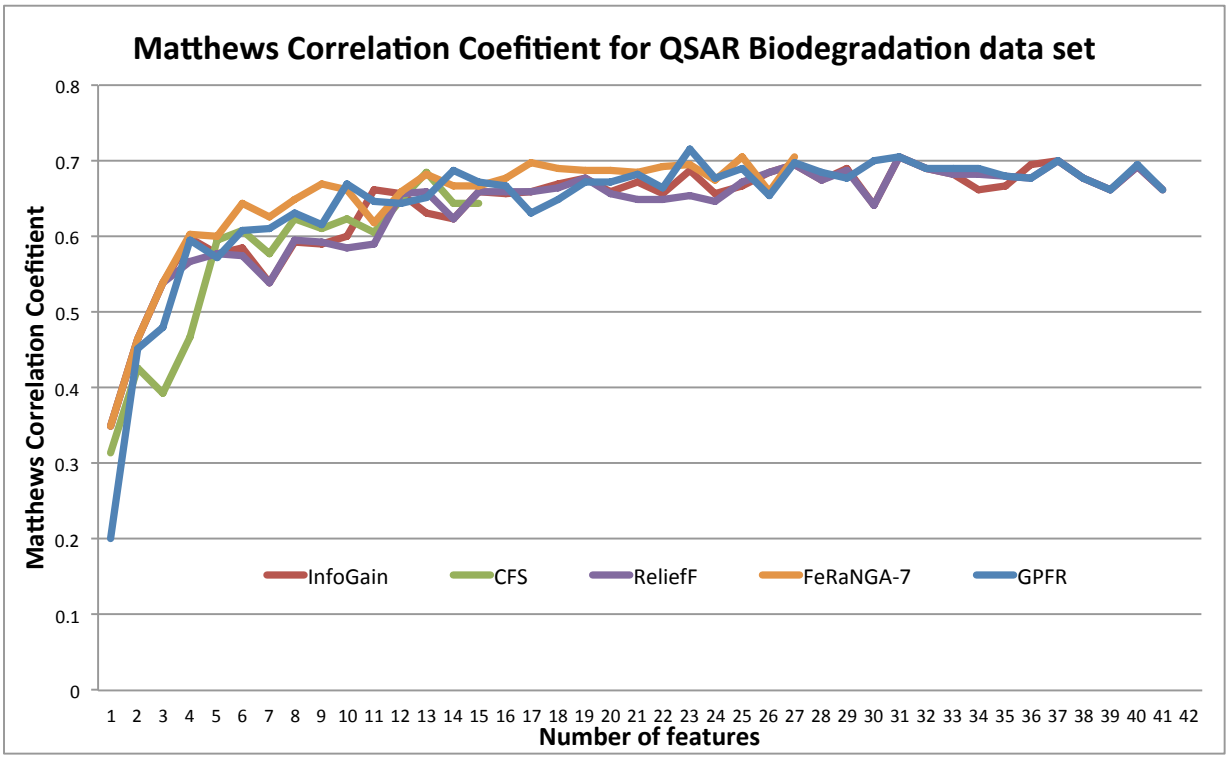

Figure A.1: Results of the Matthews correlation coefficient (MCC) for FeRaNGA-n and GPFR methods on QSAR Biodegradation data set in comparison with several state-of-the-art methods results. The FeRaNGA-n ranks were obtained for the NGA settings of 150 individuals in population and 150 epochs. For MCC computation, the attributes were ordered from the best one on the left to the worst one on the right. The best MCC was achieved for the GPFR method, the second best result was for FeRaNGA-n7.

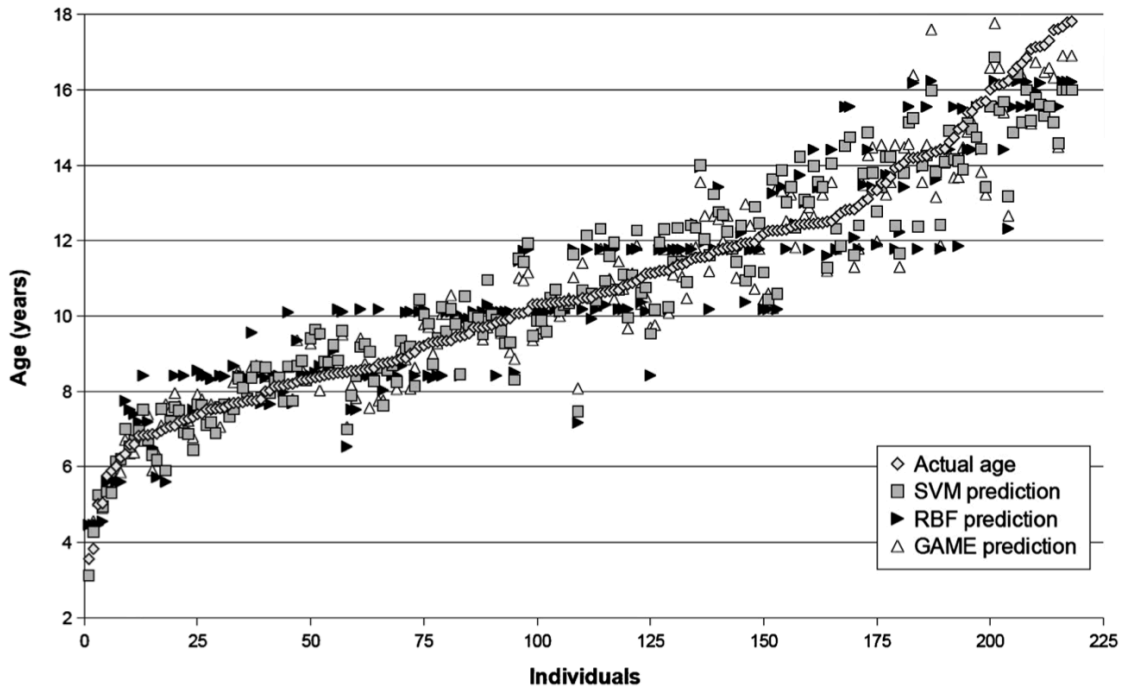

Figure A.2:]
Graphic comparison of the chronological age of 220 randomly selected girls and their age estimation using the most successful predictive models.

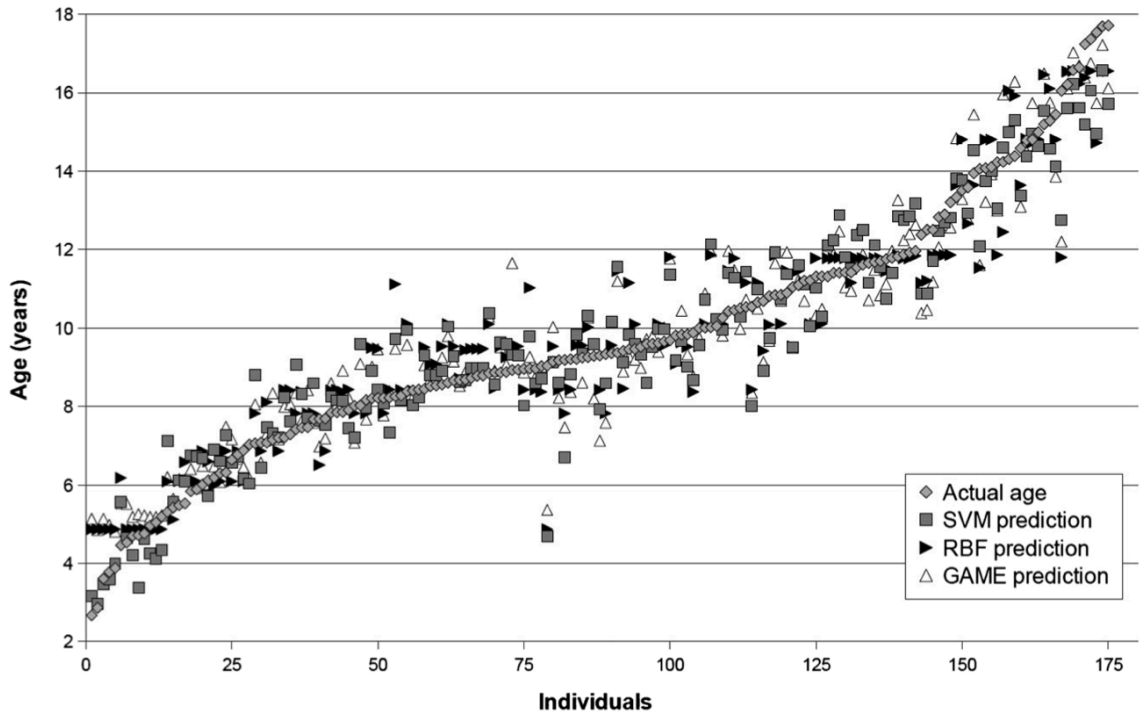

Figure A.3: Graphic comparison of the chronological age of 170 randomly selected boys and their age estimation using the most successful predictive models.

Table A.3: Different NGA configuration of GAME for FeRaNGA results on the Housing data set. Gray background represents not selected attribute. There is obvious, that the most important attribute is the attribute nr. 1. Configuration of the GAME algorithm consists of number of epochs (first number) and size of initial population.

Configuration	Ranks - FeRaNGA-5 method											
30 30	1	4	5	7	2	3	6	8	9	10	11	12
50 30	1	6	7	5	2	3	4	8	9	10	11	12
75 30	1	6	4	2	3	5	7	8	9	10	11	12
100 30	1	6	7	2	3	4	5	8	9	10	11	12
150 30	1	2	3	4	5	6	7	8	9	10	11	12
300 30	1	5	6	4	7	10	8	2	3	9	11	12

Appendix B

Lists of abbreviations

- ANN ... artificial neural network
- ASM ... adjusted stability measure
- AI ... artificial intelligence
- BC ... Borda count
- CA ... classification accuracy
- CV ... cross validation
- CR ... external criterion of regularity
- DC ... deterministic crowding
- DM ... data mining
- DMM ... data mining methods
- EA ... evolutionary algorithm
- EC ... evolutionary computation
- FAKE-GAME ... fully automated knowledge extraction - group of methods data handling
- FE ... feature extraction

- FeRaNGA . feature ranking based on niching genetic algorithm
- FR ... feature ranking
- FS ... feature selection
- GA ... genetic algorithm
- GAME ... group adaptive methods evolution
- GMDH ... group method data handling
- GP ... genetic programming
- GPFR ... genetic programming based feature ranking
- GPRS ... genetic programming based feature selection
- KDD ... knowledge discovery from databases
- MCC ... Matthews correlation coefficient
- MIA ... multi-layered iterative algorithm
- ML ... machine learning
- NGA ... niching genetic algorithm
- NN ... neural network
- UNC ... unique chromosome

Appendix C

Acronyms and symbols

All acronyms are defined when first used in the text, with the exception of frequently used ones.