University of Modena and Reggio Emilia in agreement with

University of Ferrara

and University of Parma

PhD in Mathematics - Cycle XXXIV

New Mathematical and Computer Science Methods for the Optimization of the Study and the Exploitation of Natural Resources

Coordinator:
Prof. Giardinà Cristian

Author:
Lucena Sánchez Estrella

Supervisor:
Prof. Sciavicco Guido

Co-Supervisor:
Prof. Vaccaro Carmela

February, 2022

Contents

Preface

During the three years of my PhD research, I worked together with my supervisor, Prof. Guido Sciavicco on many different projects. Although some of these projects brought us relevant publications not included in this thesis, they are still worth mentioning. In one instance [1], we proposed a new method for *symbolic* multivariate temporal regression which was then applied to improve our previous results in the air pollution problem (Chapter 6). In another publication [2], we provided a novel *discretization* algorithm to perform an association rule extraction that was also applied to the same data. In a third instance [3], we designed a standard prediction model for specific oceanic resources. In all of these publications, we worked in the subfield of explainable machine learning and we applied these novel models to solve real-world problems. This thesis focuses on solving real-world problems by providing explanatory and interpretable results. The difference between the results in these three publications and those included in the thesis is that the former publications entailed working on new algorithms at a much lower level when compared with the latter publications (in [1, 2]), and their results are certainly less immediately applicable, or working with standard technologies (in [3]). The results included in this thesis, on the other hand, are characterized by a new methodology (proposed in this thesis) combined with standard learning algorithms.

Chapter 1

Introduction

1.1 A General Framework

Artificial intelligence, and particularly machine learning, has been the most powerful tool for intelligently automating human activities in the last decades. In different fields, artificial intelligence helps people and solves real-world problems. It is a tool that, by helping the understanding of natural processes, brings new solutions to common and unresolved problems. In present times, mathematics and computer science work together to solve problems, to create new problems and to discover new possibilities and approaches. This is the premise of this research: to develop new mathematical and computer science methods that provide better solutions to real-world problems by searching for optimal models. Specifically, we studied preprocessing and machine learning techniques to obtain better and more relevant results than those existing in current literature. Our goals were also to obtain interpretable and explainable results and to find the optimal model according to the application specificity. In order to develop the best models and, later, a general mathematical framework that will allow us to solve various real-world problems, we first needed to study real data and its characteristics. The issues studied in this research concern the conservation and preservation of natural resources such as water, as well as earth and atmospheric resources. The current approaches often consist of long-term (sometimes *by hand*) processes to manage huge amounts of information and associated data. We developed methods that learn from these data and then used them to better understand the natural processes. As examples, we approached the problem of the protection of water resources by concerning the preservation of the availability of fresh water; we studied the characterization of medieval buildings; and we studied the problem of recognising the factors that have the greatest impact on the concentration of air pollutants at a specific time and geographical location. After analyzing the data, we realized that all of the issues above had commonalities: the data consists of a set of chemical, physical, and/or isotopic origins, they require similar data preprocessing steps, and they are meant to produce interpretable models and results. Thus, we need an initial preprocessing phase, a learning algorithm to solve specific problems, and interpretability measures. Given this, the methods that we are looking for must take into account *both* the performance of the learning algorithm and the interpretation of the results. The solutions are associated with more than one objective which, therefore, implies that our problems are intrinsically multi-objective. In machine learning, the techniques used to solve multi-objective problems are called *multi-objective optimization algorithms*. Interpretable results are results that consist of a reduced number of attributes and relations between them, so that the expert is able to give straightforward explanations, if possible. This meant that all of our problems required an optimized feature selection. This then inspired us to utilize *feature selection*, seen as a multi-objective optimization problem, as the basis to develop all of our models. In addition to providing new machine learning models to solve the real problems that we considered, this

study analyzed the properties that are common across these models, in order to obtain a more general mathematical framework. Such a framework consists of the previously mentioned elements: data input, preprocessing techniques, learning algorithms and optimization algorithms. The question that we posed is: is there a common scheme or approach to all of our solutions? Can we devise a general mathematical framework of which all our models are particular cases? To answer these questions we analyzed the building elements of each model and realized that the same procedure was repeated each time. Inspired by the feature selection process seen as a multi-objective optimization problem (well known in the machine learning community and literature), we modeled *all* of our problems as multi-objective optimization problems. As a result, we instantiated the multi-objective optimization equation with different objective functions associated with the specific problem requirements. Then, we used the individual (or gene) definition to create the new interpretability measures. The individual definition is a component of the optimization problem and we modeled it to perform the preprocessing technique. That is, we defined the individual based on preprocessing requirements and later, we used the individual representation to perform preprocessing techniques and to define the interpretability measures. Interpretability measures are the objective functions of our problem and, as previously mentioned, we used them together with the objective functions associated with the performance of the learning algorithm to instantiate the multi-objective optimization equation. In response, we provided a general mathematical framework that included data input, a preprocessing phase, a learning phase and an optimization phase, which all pivot on their common element: the *individual*.

1.2 The Framework as a Wrapper

Feature selection in real cases such as ours is used to determine exactly which independent variables have a role in learning certain tasks. Feature selection methods (see Figure 1.1) are categorized into filters, wrappers and embedded models. *Filters* are algorithms that perform the selection of features using a statistical measure that classifies their significance without making use of any machine learning algorithm; *wrappers* evaluate the attributes driven by the performances of an associated learning algorithm, and *embedded models* perform the two operations (selecting variables and building a classifier) at the same time. While embedded models tend to produce very good subsets efficiently, they are usually complex and difficult to implement. On the other hand, filters are very simple, and their results are often of lower quality when compared to wrappers. We used the idea of feature selection as a wrapper (see Figure 1.2) as the basis of all our implementations as this allowed us to separate the various components, and focus on each of them separately. Regarding the optimization of algorithms, the range of possible choices is very wide. In terms of purely multi-objective optimization, though, the possibilities are reduced; our option was to use *genetic algorithms* for this role. Exploring what happens when genetic algorithms are substituted with other paradigms in the same context is a open problem, both in terms of definitions and in terms of results. A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring for the next generation. The process of natural selection starts with the selection of the fittest individuals from a population. They then produce offspring which inherit the characteristics of the parents, which will continue to the next generation. If parents have better fitness, their offspring will be better than the parents and, therefore, have a better chance at surviving. This process keeps on repeating and will result in a generation with the fittest individuals. This notion can be applied to a search problem, where we consider a set of solutions for a problem and then select the set of best ones. In a genetic algorithm, there are several important components, which include, the choice of the initial population; the choice of the fitness function(s); and the definitions of three operators, namely selection, crossover and mutation. The process begins with a set of individuals

known as the *initial population*. Each individual is a solution to the problem that we want to solve. An individual is characterized by a set of parameters (variables) known as *genes*. Genes are joined together as a string to form a *chromosome* (i.e., a solution, or individual) and usually, binary values are used to encode it. In our framework, we codify the genes of an individual depending on the preprocessing requirements of the specific problem. That is, we use different values (binary, integer or real, or ranged values) and we define the size of an individual based on the preprocessing technique necessary for the application. For example, to perform the classical feature selection, we use binary values and we set the individual size as the number of attributes of the data set. For example, a gene with a value of 0 means that the attribute in that gene position is removed and a gene with a value 1 means that the attribute in this gene position is selected. As another example, to perform an outlier detection, we set gene size as the number of instances of the data set and we used binary values whereby 0 means that the instance is deleted and a 1 value means that the instance is preserved. Accordingly, let us observe how interpretability measures can be defined by using this individual representation. We can define the *cardinality* of an individual as the number of features selected by the feature selection process. Considering the previous definition of our individual in the feature selection process, cardinality is defined as the number of values 1 of the individual itself. The cardinality function can be considered as an objective function and can be minimized in the optimization process. We shall see that similar measures can be defined in different interpretations of the individual in order to solve different problems. The fitness function is an evaluation function that determines how fit an individual is (i.e. the ability of an individual to compete with other individuals) by giving a fitness score to each individual. This fitness score, in our case, is associated with the performance of the learning algorithm of a particular solution as well as the interpretability of such solution. For example, if the learning algorithm consists of a linear regression, one fitness function may be the correlation coefficient of the regression model. A fitness function is considered as an objective function, and will be maximized or minimized in the context of the optimization, depending on the case. Currently, we have a population whose individuals have a fitness score that determines their level of fitness. However, a population usually consists of a large number of individuals and, given this, the goal of the next phase is to gather the individuals with the highest fitness scores. The main goal of the *selection* operator is to make duplicates of the good solutions and eliminate the bad solutions in a population, while keeping the population size constant. The selection operator characterizes a specific genetic algorithm over the others. Our choice was to use a well-known algorithm, whose suitability is accepted in the literature; we did not explore different options. The *crossover* operator(s) make their selections from the mating pool at random, and sections of the strings that represent the individuals are exchanged between them to create two new strings. We used the *single-point* crossover operator that performs by randomly choosing a crossing site along the string and by exchanging all bits on the right side of the crossing site. Here, depending on the individual representation, we defined the range of the crossing sites; in other words, we created different versions of this operator for different problems. This operator is mainly responsible for the search aspect of genetic algorithms, even though the mutation operator can also be used for this purpose. The *mutation* operator consists of randomly flipping some of the sections of the strings that represent individuals, with some form of a mutation probability. In those cases in which individuals are represented with non-binary pieces, the mutation had to be redefined. We limited our versions of the mutation by setting a range of possible (and meaningful) values. Finally, the typical termination condition consists of checking if the population has converged (does not produce offspring which are significantly different from the previous generation) or a predetermined number of iterations has been reached.

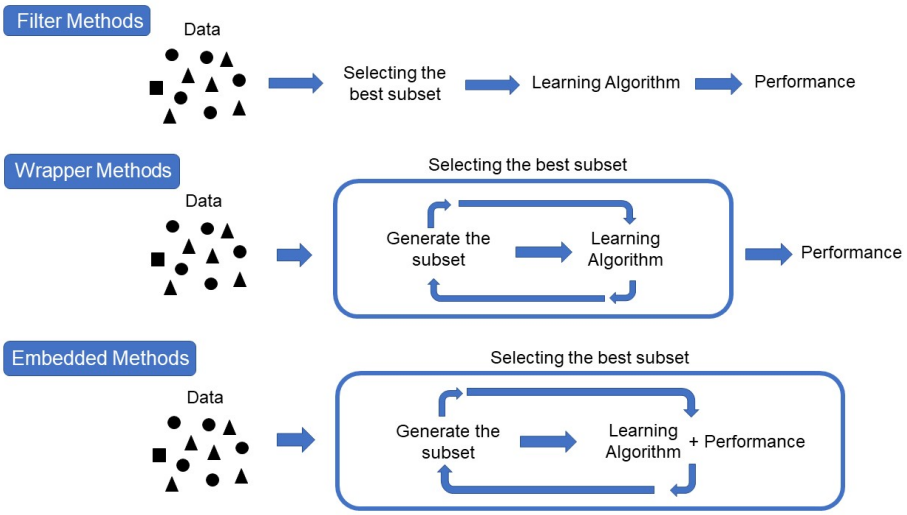


Figure 1.1: Feature selection methods.

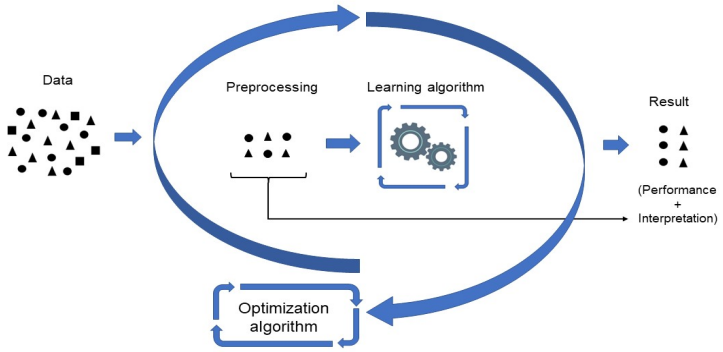


Figure 1.2: Wrapper.

1.3 The Framework as a Mathematical Object

The final goal of this thesis is to define a general framework in an abstract sense, in addition to the practical and implementation one. We aim to design a general form of operator that applies to classically presented data, i.e., in tabular form, by taking into account their components: instances, attributes, and dimensional characteristics. We are interested in problems with final solutions that will be the result of feature selections, outlier detections, and feature extraction and transformation. Inspired by the wrapper-based approach to feature selection, we defined all such processes as wrappers and, in the end, as particular cases of the same mathematical operator acts on the data. Such an operator is parameterized by the specific preprocessing that the data require, and the parameterization determines the shape and the interpretation of an individual (in terms of multi-objective optimization solved via a genetic algorithm). By abstracting all such processes, our purpose is to study the properties of this operator and, in due course, to lay down the bases for their integrated treatment. In this sense, we shall define very practical objects such as *water fingerprints* or *geochemical characterization* as, depending on the case, specific applications of the same, general operator, therefore offering a clean, mathematically well-founded characterization of processes that in the literature are, in many cases, simply applied (that is, without any formal treatment or interpretation).

Structure of the Thesis

The thesis is organized as follows.

- In Chapter 1 we present the premise of this work and the real-world problems studied. Then, we explain the steps we followed to obtain the general mathematical framework definition. We introduce the more important elements and properties of the framework and finally, we show the math application nature of our framework.
- In Chapter 2 we give the necessary background of the components of our general framework. We deal with data type, data preprocessing techniques, data analysis techniques and optimization algorithms.
- In Chapter 3 we describe our methodology. First, we present the math characterization of our general framework. We include math characterization of data and the prepocessing techniques used, individual and the iteration between them. Finally, we present the implementation by using the optimization algorithm.
- In Chapter 4 we present a problem related to protection of water resources, concerning the preservation of the availability of fresh water. We explain models used to solve it and the results obtained.
- In Chapter 5 we present the atmospheric resource problem. We explain the models that have been used to solve it and the results obtained.
- In Chapter 6 we explain the problem associated to earth resource. We describe the model used to solve it and the results obtained.
- Finally, in Chapter 7 we mention the main objectives satisfied by this thesis, our contributions, as well as the remaining questions to be addressed in future work.

Chapter 2

Background

2.1 Data

2.1.1 Data Sets

Definition 1 (data set as a collection) *A* data set $\mathcal{I} = \{I_1,\ldots,I_m\}$ *is a finite collection of* m *instances, each of which is characterized by values of* n *distinct attributes* $\mathcal{A} = \{A_1,\ldots,A_n\}$ *and a name,* N*. We say that* $\mathcal{I}$ *is* labeled *if every instance* I *is associated to exactly one class* $C \in \mathcal{C} = \{C_1,\ldots,C_k\}$*.*

A data set may contain spatial or temporal information, which means that some of the attributes may be temporal or spatial coordinates; this is a very general setting that captures how most data sets are traditionally presented. Needless to say, non-spatial and non-temporal data do not have any coordinates. In an ideal situation, spatial coordinates are assumed to be codified in a single column, while in reality, they are usually spread out on several dimensions. When possible, names are used to identify the different devices, persons or resources from which the features are collected. Thus, in the simplest case, data can be collected from a single source, without any spatial-temporal identification. While in the most complex cases, we have several sources (therefore, several names), each of which contributes data at different times and locations. Classes may be categorical or numerical and we shall see how theses cases are associated with problems that, although, are different in nature, they may be approached in very similar ways. By considering the instances of a data set as *rows* and the attributes as *columns*, a data set can be defined as a matrix, as follows.

Definition 2 (data set as a matrix) *A* data set $\mathcal{I}$ *is a matrix with* m *rows and* $n+4$ *columns* $\mathcal{A} = \{N, A_1,\ldots,A_n,T,S,C\}$*. The columns* $A_1,\ldots,A_n$ *contain the values of* n *independent attributes, while the column* N *(resp.,* T,S*) contains the values of the name (resp, temporal, spatial) attributes:*

$$\mathcal{I} = \begin{bmatrix} N_1 & a_{11} & a_{12} & \ldots & a_{1n-4} & t_1 & s_1 & c_1 \\ N_2 & a_{21} & a_{22} & \ldots & a_{2n-4} & t_2 & s_2 & c_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ N_z & a_{m1} & a_{m2} & \ldots & a_{mn-4} & t_m & s_m & c_k \end{bmatrix}_{mn} \tag{2.1}$$

where $z, k \leq m$*.*

In the rest of this work, we shall use several alternative notations and we shall make some simplifying assumptions. For a given attribute A, we shall use $dom(A)$ to denote its domain, that is, the set of all values that occur in the column A in $\mathcal{I}$. We assume that $dom(T)$ is an initial segment of $\mathbb{N}$. As such,

this assumption allows us to deal with most real-world cases. It should be noted that we are, in fact, discarding temporal distances between events. We make the same assumption for $dom(S)$; even if space can be generally described by several coordinates with finite domains, one can always encode tuples in single numbers. In this case, $\mathcal{I}$ may be associated with one or several auxiliary functions and we shall always assume that there is a function $dist(s, s')$ that returns the distance between two spatial locations s and s' along any of the coordinates.

2.1.2 Knowledge Extraction

Historically, different approaches have been used to extract information from a data set. In the simplest cases, names, time, space, and classes can be ignored, and extracting knowledge from a data set $\mathcal{I}$ is, therefore, reduced to understanding the mathematical behavior of the attributes, often referred to as *descriptive* statistics. In doing so, we analyze each attribute (or variable) independently, and the goal is to summarize the information that we can extract from such an analysis. We use descriptive statistics to describe individual quantitative observations and, very often, the methods that are applied are graphical by nature (i.e. histograms, boxplots, and scatter plots). Descriptive statistics are progressively integrated with *inferential* statistics when relations between variables are taken into account, and even in simple scenarios, much effort can be devoted to understanding the possible multivariate relationships that the data present. As indicated previously, such relationships may help to understand how the data behave. When considering classes, the idea of *causality* arises, and *regression and classification* models are used to understand the processes that underlie the data. Regression is the term commonly used to denote the inferential methods for describing the behavior of one variable (for this study, the class variable when it is numeric) in terms of other variables (for this study, the attribute variables). Classification, on the other hand, deals with non-numerical classes, and consists of describing how the class variables depend on the values of the attributes. These two problems originally stem from different areas (inferential statistics the former, *machine learning* the latter) but share a number of concepts and methods. It can be argued that, as of late, machine learning when intended as a discipline, includes both regression and classification tasks and algorithms. Moreover, the ever-growing quantity of data that are produced and recorded, and that are therefore object of analysis, justifies a *frequentist* approach to learning – typically associated with machine learning, as opposed to a *distribution-based* – typically associated to inferential statistics. Interestingly, focusing on classification does not automatically entail that neither descriptive or simple inferential methods become ineffectual. For example, a simple binary classification problem can be approached by observing how the mean of a particular attribute changes among positive and negative cases, and a naïve classifier can be designed based on such an observation. The same principle applies to any other descriptive method, as well as to non-class oriented inferential methods (e.g., one could build a classifier based on the level of linear correlation between two columns). Such immediate approaches can be evaluated using the same machinery employed for true classifiers, that is, exploiting concepts such as *training+test* approaches, among many others. In a sense, pure classification and regression approaches generalize such simple approaches. In the case of real-world data and problems, however, classification and regression may not be enough. As a matter of fact, in real-world situations, the premise is not necessarily searching for relations between variables and/or the class, and ultimately, many practical problems cannot be solved by simply classifying or predicting something. *Meta-learning* can be defined as a set of techniques that are independent of the learning algorithm and that, when applied, help the task of the learning algorithm. These techniques are classically separated into *feature selection and transformations*, *instance selection* and *outlier detection*. Feature selection is defined as the process of selecting features from the database that are relevant to the task to be performed [4]. Moreover, feature selection can be generalized to feature transformation in order to take into account temporal and spatial aspects (not usually considered by conventional classification and regression algorithms) as well

as non-linear behaviors of single variables (up to a limited point). Instance selection is defined in [5] as the process that selects or searches for a representative portion of data that can fulfill a data knowledge discovery task as if the whole data is used. Finally, outlier detection is defined in [6] as the process of eliminating samples from the data base that do not comply with the general behavior of the data model. All together, meta-learning techniques are used in combination with conventional classification and regression algorithms. Their aim is not only to ease the task of the former, but also to extract more abstract information that transcends the classification/regression algorithms and is, ultimately, the solution to a practical problem. The key observation is that all such meta-processes share several aspects: they consist of specific kinds of matrix transformations, they can be defined as optimization problems, and they are essentially independent from the used classification/regression approach. Their versatility makes meta-learning methods the ideal solution to a great variety of problems because they can be immediately adapted to the multi-names cases, and they can be all defined as particular cases of the same operator, which we call here, the *dynamic preprocessing* operator.

2.2 Canonical and Non-Canonical Preprocessing

2.2.1 Feature Selection

Feature selection (FS) is defined as the process of selecting features from the database that are relevant to the task to be performed [4]. Feature selection facilitates data understanding, reduces storage requirements, and lowers processing time, so that model learning can become an easier process. Feature selection methods that do not incorporate dependencies between attributes are called *univariate* methods. They consist of applying specific criterion to each pair feature-response, and measuring the individual power of a given feature with respect to the response independently from the other features, so that each feature can be ranked accordingly. In *multivariate* FS, on the other hand, the assessment is performed for subsets of features rather than single features. In many cases, the data set will be better characterized by the sole features with highest discriminative power than by the whole features, because the irrelevant and redundant ones should be regarded as noise. However, when an individual feature in a high dimensional feature, space presents small correlation with the target class, it may be still useful in combination with other features. In this regard, multivariate FS tends to give better results than univariate FS [7]. From the evaluation strategy point of view, FS can be implemented as a *single attribute evaluation* (in both the univariate and the multivariate case), or as a *subset evaluation* (only in the multivariate case). Feature selection algorithms are also categorized into filter, wrapper and embedded models (for example, *lasso regression* [8] in regression models). Wrapper methods for feature selection are more common in the literature and are often implemented by defining the selection as a search problem, and solved using metaheuristics such as evolutionary computation (see, e.g., [9,10]). Wrapper schemata are more common in supervised a classification than in a regression however, some effort in this direction has been made (see, e.g., [11]). Other strategies for FS include *adaptive boosting* [12].

2.2.2 Feature Transformation: the Temporal Case

Traditionally, in machine learning, methods that handle temporal information are the same as those that handle time series. A *time series* is a series of data (in our case, a row of the matrix) labelled with a temporal stamp. In the literature, there are two main problems associated with time series: *time series explanation* and *time series forecasting*. These problems are usually associated with different contexts and are approached with different tools, which, however, share some common ideas. Among these, *lagged models* are the best known ones. In lagged models, data are systematically transformed by adding a delay so that a traditional, propositional learning algorithm can be then applied; among the available packages

for this purpose, we reference WEKA's *timeseriesForecasting* [13]. Lagged models are flexible by nature because they are not linked to a specific learning schema, and their focus is on time series explanation. While explicit models can be used for forecasting, it is not their focus considering that their forecasting horizons are limited to the maximum lag in the model. The key point to note is that lagged variables can be seen as a form of feature transformation, in which the correct lag is chosen (i.e. delay) to apply to a certain column (possibly, within a certain name). Given this, we could argue that feature selection is a particular form of *lag selection*. Such an observation can be further generalized into considering a combination of different, but temporally adjacent, lags for the same variable, driven by a linear function, as an example. As another example, consider temporal data for an air pollution problem and assume that, among several variables, one particular variable denotes the wind speed at a certain location at a certain time. While it is clear that the wind speed influences the pollution concentration at that same location, the temporal aspects of such an influence may vary. To take this aspect into account, we may consider a data transformation that combines different lags (e.g., the wind speed two hours before the considered moment, one hour before, and at the considered moment), and assume that they all have an influence. By combining such lagged variables into one, we may obtain a more informative, and very interpretable data set to be used in a regression problem, for instance. We call this type of transformation *lag selection with intervals and functions*.

2.2.3 Feature Transformation: the Spatial Case

In a similar way, when working with spatial data, each observation (i.e. each row of the data matrix) is associated with a multi-dimensional location. In the planar case, for example, each location may be represented with two Cartesian coordinates (for simplicity, these are denoted as a single column S in Definition 2). The spatial equivalent of lagged data in the temporal case is known as *flatted data*, and consists of creating, whenever necessary and convenient, new variables that contain the value of the original ones at neighboring locations. We interpret this as another form of feature transformation, and we treat it similar to the temporal case. Not only does this kind of spatial transformation entail the problem of establishing distance, and in which direction(s) one should go per each point but also, different locations may be combined into single variables as we did for different lags. An example of this application can be seen when considering the problem of studying the presence of a certain biological entity at each point of the area under study, at a certain moment in time. It may the case that the variables, whose influence we are assessing, exercise such an influence from the neighboring locations at earlier times. We call this transformation *spatial selection with intervals and functions*. Again, going after a higher interpretability, both the area of influence and the ways in which the attribute influences are limited. Not unlike in the temporal case, but with higher degrees of variability, spatial transformations may be performed by searching into different shapes of neighboring areas, and combinations may use different functions. Although the principles are the same, in our experiments we will limit ourselves to simple, squared areas and linear combinations.

2.2.4 Feature Transformation: the Nonlinear Case

Finally, feature transformations allow us to deal with nonlinear problems, on a limited basis, which arise in systems in which the change of the output is not proportional to the change of the input. Nonlinear problems are of interest to the fields of engineering, biology, physics, among many others, because most systems are inherently nonlinear in nature. Traditional methods of solving these problems are generally known to fall in the field of *non linear programming*. But, to a certain extent, nonlinear transformations are feature transformations and can be treated as such. We limit ourselves to considering simple nonlinear transformations in which variables are simply elevated to a certain power. Nonlinear

transformations, however, can be combined with both lag selection and spatial selection, whenever necessary and convenient.

2.2.5 Outlier Detection

Outlier detection (OD) is defined in [6] as the process of eliminating samples from the database that do not comply with the general behavior of the data model. Such samples, called *outliers*, differ significantly from the remaining records in the data set, and their existence can be either caused by measurement error or they may be the result of inherent data variability. Outliers can be detected via non-algorithmic methods (for example, as seen in in [14]) however, more recently, algorithmic methods are the preferred method of detection. While OD methods can be classified following a taxonomy similar to the one used for FS methods, in the literature, OD is first separated into supervised and unsupervised methods. *Supervised* OD defines outlier detection as a classification process. It requires a previous labeling of those instances that are known to be outliers, and then it focuses on training a classifier to recognize further outliers. Examples of supervised methods, which are focused on anomalous, unexpected, or fraudulent behavior detecting include [15–19]. The most relevant difference between supervised OD and a simple supervised classification problem is derived from the intrinsic unbalancing of data in the latter case. When referring to generic outlier detection, *unsupervised* OD, which is more common than a supervised OD, is further separated into *density-* (see, e.g., [20] and many other variants of the algorithm LOF) and *distance*-based methods (see, e.g. [21, 22]). In both cases, these methods treat the OD problem as a sort of clustering problem. In our terminology, all these methods can be classified as filters (unsupervised and supervised, univariate and multivariate), as the process of detecting outliers is not linked to any successive learning task or algorithm. Univariate, unsupervised filters for OD are also the most commonly available methods. For example, the open-source suite Weka offers the filter *InterquartileRange*, which allows for the detection of outliers and extreme values by looking at the values of any attribute, under a normal distribution hypothesis. Embedded models for OD in regression have been attempted. In [23], for example, multiple outliers are detected during the learning process, and in [24, 25] a genetic programming-based approach to outlier detection has been presented. A comprehensive comparison among several OD methods can be found in [26]. Outlier detection methods can be also classified as filters, wrappers, and embedded methods just as FS approaches are. However, unlike FS, this classification is not widespread. In the case of a regression problem there are, however, notable embedded OD methods. An example is *robust regression* (see, e.g., [27]), which is a portfolio of regression techniques that are designed to respond better than classic regression techniques when the usual hypothesis on which they are based fail, as can be seen in the presence of outliers. Therefore, they should be listed as outlier detection techniques. Robust methods are based either on substituting the least square principle with other principles, such as *least absolute deviation*, *m-estimation*, *least trimmed squares*, or on replacing the normal distribution with a *t-distribution*, or on using *unit weights*. Robust regression is not as popular as classical regression, algorithms are not very widely present in learning and statistical suits, and they have been often criticized for being computationally demanding. Robust regression has also been approached via evolutionary algorithms [28].

2.2.6 Instance Selection

Instance selection (IS) is defined in [5] as the process that selects or searches for a representative portion of data that can fulfill a knowledge extraction task, such as classification or regression, as if the whole data was used. Utilizing IS reduces an original data set to a manageable volume, which leads to a reduction in the computational resources needed to complete the learning process. As in FS and OD [29], IS methods can be divided into filters, wrappers, and embedded methods. Instance selection methods

use information measures or rules to evaluate instance subsets, such as *weakness* [30], *clustering* [31], *weighting* [32], *relevance* [33], and so on. Instance selection wrapper methods use a classifier to evaluate candidate instance subsets. Most of the IS wrapper methods have been proposed based on the $k-NN$ classifier [34], and this type of instance selection is called *prototype selection* [35]. Prototype selection methods attempt to select a representative subset of samples from the training set, while *prototype generation* methods [36] generate a small set of artificial prototypes to replace the original training set. These methods are usually based on *misclassification* [37–39], although there are others based on *associate sets* [40] or on *support vector machines* [41]. As in FS, IS can be considered as a search problem, and *forward*, *backward* and *mixed* search strategies can be applied for IS [42], in addition to metaheuristics such as evolutionary algorithms. Finally, IS methods can also be categorized according to the *selection strategy* [43] into *condensation algorithms* that attempt to remove the instances far away from the decision surface, *edition algorithms* that remove noises to improve the classification accuracy, and *hybrid algorithms* that remove both central and border instances, and a reduced set can be obtained by merging border and core instances [44]. Instance selection is not only connected to classification tasks, but also to regression tasks [45–47]. Overviews of IS methods can be found in [48–50]. Just as FS and OD, IS tasks can be also combined freely with lag as spatial selection, when necessary, as well as with nonlinear selection.

2.2.7 Combined Models

Combined models refer to applying FS and OD or FS and IS to the same data. Combined models range from *sequential* combinations (e.g. FS+IS versus IS+FS), to *simultaneous* models, although the latter are not usually defined as optimization models. Combined FS and OD has been attempted in the literature using several different schemata. One of the early approaches includes [51], in which a two-step, rather primitive, technique for multivariate simultaneous FS and OD was presented. A more recent diagnostic method, based on dummy variables and evaluation-by-plotting specific for linear regression, has been presented in [52], and a two-step method specific for linear regression, in [53]. Combined models also include [54], based on a neural network trained by an evolutionary algorithm, as well as [55–58], which are specific for combining FS with IS. In general, the simultaneous strategies have shown the highest performance.

2.3 Canonical Learning Algorithms

Machine learning includes numerous learning algorithms that depend on the data type and the task to be performed. Learning algorithms can either be *supervised* or *unsupervised*. The difference between these two main classes is the existence of labels in the training data subset. According to [59], supervised machine learning involves a predetermined output attribute in addition to the use of input attributes. The algorithms' attempt to predict and classify the predetermined attribute, and their accuracies and misclassification alongside other performance measures, is dependent upon the ability of the predetermined attribute to correctly predict or classify or otherwise. Accordingly, supervised learning can be divided into two categories: *classification* and *regression*. Conversely, unsupervised data learning involves pattern recognition without the involvement of a target attribute. That is, all the variables used in the analysis are used as inputs and because of the approach, the techniques are suitable for *clustering* and *association* mining techniques. Classification and regression models can be broadly divided into *tree-based*, *function-based* and *rule-based*, depending on how the model is represented. Tree-based classification models are characterized by describing the underlying theory as a tree, and they range from *deterministic single-tree* models, such as ID3 or C4.5, to *random tree* models, to *random forest* models (see, e.g., [60–62]). Function-based classification models range from the very simple *logistic regression* models, to *linear regression* models, to *neural network* models (see, among others, [63–65]), and they are

Algorithm type	Category	Name	Source
Classifier	Tree-based	j48	*Weka*
		Random Forest	*Weka*
Classifier	Function-based	Linear regression	*Weka*
		Logistic regression	*Weka*
		Sequential Minimal Optimization	*Weka*
		Multilayer Perceptron	*Weka*
Cluster	Centroid-based	Simple K Means	*Weka*
MOEA	Elitist Non-Dominant	NSGA-II	*jMetal*

Table 2.1: Summary of machine learning algorithms used.

characterized by the fact that the underlying theory is modelled as a function where the output value is used to determine the class. Finally, rule-based classification models describe an underlying theory by means of sets of rules, which are independent from each other and, in a sense, imitate the human reasoning (see, among others, [66–68]). Rule-based classification generalizes tree-based classification, since a tree can be seen as a particular case of rules set, but not the other way around. Moreover, single trees and rule-based models are generally *interpretable*, that is, they are models that can be read (and explained) by a human, while function-based models (especially those based on neural networks) are not. In this thesis, we focus on tree-based and function-based algorithms for classification and regression, as well as on clustering algorithms. All of these are seen as a *black box* that is independent of the preprocessing technique. From a strict implementation point of view, we use Java and, in particular, the classes *classifiers* and *clusterers* from the open-source learning suite *Weka* [69]. we have also implemented some other derivated models using the *eclipse* environment. As classifiers of Weka, it was used in the *linearRegression*, *logistic*, *SMO*, *multilayerPerceptron*, *randomForest*, *j48* and *REPTree* algorithms. As clusterers of Weka, we have used the *simpleKMeans* algorithm. A summary of this is shown in Table 2.1, which also includes the evolutionary algorithm *NSGA-II* explained in next sections.

2.3.1 Tree-Based Classification and Regression Models

The *decision tree* model is probably the most widely known classification and regression model. Originally, it was studied in the fields of decision theory and statistics however, it was found to be effective in other disciplines such as data mining, machine learning, and pattern recognition. Decision trees are also implemented in many real-world applications and considering the long history and the high level of interest in this approach, several surveys on decision trees are available in the literature, such as [70, 71]. A decision tree is a classifier expressed as a recursive partition of the instance space. It consists of nodes that form a rooted tree, meaning it is a directed tree with a node called *root* that has no incoming edges. All other nodes have exactly one incoming edge. A node with outgoing edges is called an *internal* or *test* node. All other nodes are called *leaves* (also known as terminal nodes or decision nodes). In a decision tree, each internal node splits the instance space into two or more subspaces according to a certain discrete function of the input attributes values. In the simplest and most frequent case, each test considers a single attribute, such that the instance space is partitioned according to the attribute's value. In the case of numeric attributes, the condition refers to a range. Each leaf is assigned to one class representing the most appropriate target value. Alternatively, the leaf may hold a probability vector indicating the probability of the target value having a certain value. Instances are classified by navigating them from the root of the tree down to a leaf, according to the outcome of the tests along the path. Algorithms that automatically construct a decision tree from a given data set are called *decision tree inducers*. Typically,

the goal is to find the optimal decision tree by minimizing the generalization error. There are various top-down decision trees inducers such as ID3 [72], C4.5 (j48 in Weka) [73], and CART [74]. C4.5 is an evolution of ID3 which is considered as a very simple decision tree algorithm, presented by the same author [73]. ID3 uses *information gain* as splitting criteria: the growing stops when all instances belong to a single value of target feature or when the best information gain is not greater than zero. Contrary to ID3, C4.5 uses *gain ratio* as splitting criteria: the splitting ceases when the number of instances to be split is below a certain threshold, and error-based pruning is performed after the growing phase. C4.5 is capable of handling numeric attributes. It can induce from a training set that incorporates missing values by using the corrected gain ratio criteria. On the other hand, *random forest* models, seen as tree-based algorithms, are also used for classification and regression tasks. The random forest technique uses the decision tree model for parametrization, but it integrates a sampling technique, a subspace method, and an ensemble approach to optimize the model building. A random forest model is made up of a large number of small decision trees, called estimators, which each produce their own predictions and by combining the predictions of the estimators, produces a more accurate prediction.

2.3.2 Function-Based Classification and Regression Models

Function-based learning refers to the set of algorithms that solve or approximate a learning task via a mathematical function, as opposed to a logical function (as in tree-based methods). Whereas logical learning is usually associated with classification (but, as we have seen, is adapts perfectly to regression tasks), function-based learning is usually associated with regression, but all learned functions can als be used for classification. This is particularly true for modern function-based learning that is based on neural networks, but the argument works naturally with any learning function model. *Linear regression* is the most widely known function-based learning algorithm, and the literature on algorithms to approximate linear functions is wide and very well-known. *Logistic regression* is another widespread approach to function-based regression and, seemingly, can be seen as a particular case of *neural networks* or *artificial neural networks*. Artificial neural networks are an important tool that can adapt and adjust itself to the data, irrespective of any special functional or distributional requirement of the primary model. They are widely used for nonlinear pattern recognition and regression [75]. However, they are considered as black boxes due to the lack of transparency of internal workings and a lack of direct relevance of its structure to the problem being addressed, thereby making it difficult to gain insights. Furthermore, the structure of a neural network requires optimization which is still a challenge. Many existing structure optimization approaches require either extensive multi-stage pruning or setting subjective thresholds for pruning parameters. A *Multilayer perceptron* is a feed forward neural network that maps the inputs to an appropriate set of outputs and it is made up of several layers of nodes inside a graph, so that each layer is entirely linked to the next one with a nonlinear activation function, excluding the input nodes. Multilayer perceptron employs a supervised learning technique called *back propagation* for training purposes and a nonlinear activation function [76]. *Support vector machines* are another function-based technique that provides a classification learning model and an algorithm rather than a regression model and an algorithm [77]. This method is known as an algorithmic application of statistical learning theory and it uses some characteristics of the model and its performance on a training set to build reliable estimators from the data set. Consequently, this method can predict the model performance of an indefinite data set. Moreover, a support vector machine is able to solve a constrained quadratic optimization problem and build optimum separating borders between data sets [77]. The support vector machine can be also divided into *linear* and *non linear* models [78, 79]. It is called linear support vector machine if the data domain can be divided linearly (e.g. straight line or hyperplane) to separate the classes in the original domain. If the data domain cannot be divided linearly, and if it can be transformed to a space called the feature space where the data domain can be divided linearly to separate the classes, then it is called nonlinear support

vector machine. The *sequential minimal optimization* (SMO) algorithm has been shown to be an effective method for training support vector machines on classification tasks defined on sparse data sets. The SMO algorithm differs from most support vector machine algorithms in that it does not require a quadratic programming solver.

2.3.3 Clustering Models

Cluster analysis or *clustering* is the task of grouping a set of objects so that those in the same group, or *cluster*, are more similar to each other than to those in other groups. The literature on cluster analysis is very wide, and includes *hierarchical* clustering, *centroid-based* models, *distribution-based* models, *density* models, among many others. Centroid-based models are of particular interest for us, because they are especially useful for numerical, multi-dimensional objects such as those that arise with chemical-physical data. The concept of centroid is essential in the most well-known centroid-based clustering algorithm, that is, *k-means* [80]: given a group of objects and a notion of distance, its *centroid* is the set of values that describes an object C (which may or may not be a concrete object of the group) such that the geometric mean of the distances between C and every other element of the group is minimal. In the k-means algorithm, the groups (and even their number) are not known beforehand (this type of cluster analysis is called *exploratory*), and the algorithm is based on an initial random guess of the centroid that eventually converges to a local optimum. KNN [81] is a *distance-based* classification algorithm, whose main idea is that close-by objects can be classified in a similar way.

2.4 Optimization and Optimization Algorithms

Optimization refers to finding one or more feasible solutions that correspond to extreme values of one or more objectives [82]. Optimization can be, broadly speaking, *continuous* (*numerical*), or *discrete*. In the first case, for example, the interest is in finding solutions to mathematical functions. The applications of continuous optimization to machine learning and knowledge extraction, in general, are maximally represented by strategies such as *backpropagation* in neural networks (at the classification/regression level, rather than at the meta-problem level). Discrete optimization, in which we look for objects from a countably (in)finite set, typically integers, permutations, or graphs. It is further classified as *linear* or *nonlinear*, reflecting the fact that for linear discrete optimization, efficient algorithms exist that solve problems in an exact way. Nonlinear discrete optimization is a more difficult case, in which problems are generally characterized by being at least NP-hard. Among the several groups of approaches that can be used to solve a nonlinear discrete optimization problem, *heuristic* solutions are probably the most common ones. These are, in turn, characterized by a probabilistic approach with no formal guarantees of reaching a global optimum, but that shows good results in practice. There are several sub-categories of heuristic approaches, often associated with colorful names with the intent of suggesting the underlying ideas. They go from *tabu* searches [83], to *particle swarm optimization* [84], to *beehive optimization* [85], to *evolutionary optimization* [86] algorithms, among many others. Evolutionary algorithms have at least three notable characteristics that distinguish them from the other categories and are: they can be naturally adapted to be *multi-objective* instead of single-objective, they can be more commonly found in open-access code libraries compared with other categories, and with a superior variety in programming languages, and they have been applied far more than any other category of heuristic optimization approaches to knowledge extraction tasks, both directly and indirectly.

2.4.1 Multi-Objective Optimization

A *multi-objective optimization problem* (see, e.g. [87]) can be formally defined as the optimization problem of simultaneously minimizing (or maximizing) a set of k arbitrary functions:

$$\begin{cases} \min / \max \ f_1(\bar{x}) \\ \min / \max \ f_2(\bar{x}) \\ \dots \\ \min / \max \ f_k(\bar{x}), \end{cases} \tag{2.2}$$

where $\bar{x}$ is a vector of decision variables. Maximization and minimization problems can be reduced to each other, so that it is sufficient to consider one type only. In general, a problem of multi-objective optimization, does not have a solution that optimizes all objective functions simultaneously. For this reason, *Pareto optimal solutions*, that is, solutions that cannot be improved in any of the objectives without degrading at least one of the other ones, are considered. A set $\mathcal{F}$ of solutions for a multi-objective problem is *non dominated* (or *Pareto optimal*) only if, for each $\bar{x} \in \mathcal{F}$, there exists no $\bar{y} \in \mathcal{F}$ such that *(i)* there exists i $(1 \leq i \leq k)$ that $f_i(\bar{y})$ improves $f_i(\bar{x})$, and *(ii)* for every j, $(1 \leq j \leq k, j \neq i)$, $f_j(\bar{x})$ does not improve $f_i(\bar{y})$. In other words, a solution $\bar{x}$ *dominates* a solution $\bar{y}$ only if $\bar{x}$ is better than $\bar{y}$ in at least one objective, and it is not worse than $\bar{y}$ in the remaining objectives. We say that $\bar{x}$ is *non-dominated* only if there is not other solution that dominates it. The set of non dominated solutions from $\mathcal{F}$ is called *Pareto front*. The classical way to solve multi-objective optimization problems is to follow the preference-based approach, where a relative preference vector is used to scalarize multiple objectives. Since classical search and optimization methods use a point-by.point approach, where one solution in each iteration is modified to a different (hopefully better) solution, the outcome of using a traditional optimization method is a single optimized solution. However, the field of search and optimization has changed over the last few years due to the introduction of a number of non-classical, unorthodox and stochastic search and optimization algorithms. Of these, the *evolutionary algorithm* mimics nature's evolutionary principles to drive its search towards an optimal solution.

2.4.2 Evolutionary Multi-Objective Optimization

Evolutionary multi-objective optimization (EMO) algorithms [88] use a *population-based* approach in which more than one solution participates in an iteration and evolves a new population of solutions in each iteration. Such class of algorithms lies within the class of *genetic algorithms* (GA). Evolutionary multi-objective optimization algorithms are a good choice in solving multi-objective optimization problems because EMO problems, by nature, give rise to a set of Pareto-optimal solutions. Generally speaking, EMO algorithms can be all considered as structured as a main loop where the first population P (of solutions) is usually created at random, say $P = \{\Gamma_1, \Gamma_2, \dots, \Gamma_N\}$. At each iteration, a new population is generated by using four main operations: *(i) selection*, *(ii) crossover*, *(iii) mutation*, and *(iv) elite-preservation*. After that, the members of the population are evaluated, the selection operator chooses the best solutions with a large probability to fill an intermediate mating pool. The crossover operator randomly picks two or more solutions (parents) from the mating pool and create one or more solutions (offspring), e.g. P', by exchanging information among the parent solutions. The simplest crossover operator is the *single-point crossover* operator. Suppose there are two candidates $\Gamma_i, \Gamma_j \in P$, represented as 0-1 binary sequences of M pieces, then a randomly chosen an integer $k \in \{1, \dots, M\}$ and it generates $\Gamma'_i \in P'$ by juxtaposing the first k pieces from Γ_i with the last $M-k$ pieces from Γ_j; similarly for Γ'_j if needed. The mutation operator randomly perturbates each child, created via crossover. In this case, the most simple operator is to randomly flip a piece. The elitism operator combines the old population with the newly created population and chooses to keep best solutions from the combined population

(i.e. *survival of the fittest*). This set of operations is performed until one or more stopping conditions are met (e.g. number of generations). Biological evolution is non-deterministic by nature, therefore EMO algorithms, which resemble such evolution, are also non-deterministic, in contrast to traditional searching procedures that are deterministic. Over the past decade, a number of EMO algorithms have been suggested [82, 89–92]. One of the most popular EMO procedures is *Elitist Non-dominant Sorting GA* (NSGA-II) [93]. In our experiments, we used such an algorithm, which is available as open-source from the suite *jMetal* [94]. NGSA-II is an elitist Pareto-based multi-objective evolutionary algorithm that employs a strategy with a binary tournament selection and a rank-crowding better function, where the rank of an individual in a population is the non-domination level of the individual in the whole population.

Chapter 3

Methodology

3.1 General Mathematical Characterization

3.1.1 Notation and Definitions

As seen in the previous chapter, we are interested in a systematic exploration of data transformations in order to solve a meta-learning problem. This strategy has many advantages: by considering learning algorithms as black-boxes, we can always choose the state-of-the-art ones. By focusing on the meta-problem rather than the problem itself, we can reach a certain degree of interpretability even when we decide to use non-interpretable classificators/regressors. Additionally, by suitably transforming the data, we can deal with temporal and spatial components even when the classifiers/regressors do not offer spatial-temporal capabilities natively without resorting to the most naïve approaches such as flattening. The mathematical characterization of dynamic preprocessing allows us to decide on a clear and reasonably elegant syntax. A consequence will be that the expressive power may be slightly reduced but this, however, is not necessarily negative, considering that complex transformations may hamper the interpretability of the results. In the following, we assume that data sets are presented in their matricial form with m instances (rows), that $A_1, \ldots, A_n$ are the independent attributes, that data are distributed under z names (that is, there are z sources), and associated to k classes. Whenever necessary, we assume that $m = m^* z$, that is, that each name has m^* of the m instances associated with it. While it may be intuitive to suggest that the class should be a function of the name (that is, that different instances of the same source should be associated to the same class), in the most general setting we can give up such an assumption. There are, in fact, real-world cases in which it may not be true. Let $\mathcal{M}$ be the space of all data sets, and let $\mathcal{V} \subset \mathcal{M}$ be the space of all real vectors (i.e., single-dimension matrices).

Definition 3 *A* dynamic pre-processing operator P *is a real vector. A* dynamic pre-processing transformation *is a function:*

$$\tau : \mathcal{M} \times \mathcal{V} \to \mathcal{M} \tag{3.1}$$

that is, is the application of a dynamic preprocessing operator to a data set, that returns a data set.

From a syntactical point of view, P is generated by a simple grammar:

$$
\begin{array}{lcl}
P & ::= & P_{fst}P_{od}P_{is} \\
P_{fst} & ::= & (T,S,F)|(T,S,F)P_{fst} \\
P_{od} & ::= & B|BP_{fst} \\
P_{is} & ::= & B|BP_{fst} \\
T & ::= & N^{-} \\
S,F & ::= & N \\
B & ::= & 0|1
\end{array}
$$

where $N^{-} \in \mathbb{N} \cup \{-1\}$ and $N \in \mathbb{N}$. As the symbols suggest, the first segment is dedicated to feature selection and transformations, the second segment is dedicated to outlier detection, and the third segment is dedicated to instance selection. Each segment entails a specific interpretation of the values and such an interpretation implicitly takes care of assigning a null semantics to those P whose lengths of each part does not match the data set matrix to which is applied. This, therefore, allows us to simplify the syntactic definition of P. Our purpose is to define the semantics of τ and we shall do so by first defining the semantics of each part, and a specific transformation associated to it, which we call τ_{fst}, τ_{od}, and τ_{is}, and then combining them. In the following, if $q \in \{fst, od, is\}$, then $P_q[i]$ indicates the i-th component of P_q (a triple, when $q = fst$, a bit otherwise).

3.1.2 Semantics of Feature Selection and Transformations

Given an attribute A, then, let $A_{\mathcal{I}}(t,s)$ be the value of A at the time t and position s under a specific name in the data set $\mathcal{I}$; we use $A(t,s)$ when $\mathcal{I}$ is clear from the context. We let $A(t-t', s-s')$ be the value of A under the same name, at the time $t-t'$ and position s'', where $dist(s''-s) = s'$ is any dimension (thus, $A(t-t', s-s')$ is not uniquely defined, but, as it will be immediately clear, this is not a problem). For learning purposes, we can safely assume that if there is no information about time $t-t'$ and/or position s'' in $\mathcal{I}$, $A(t-t', s-s')$ takes a random value. Now, we use:

$$\mathcal{A}(t-t', s-s') = \{A(t-t'', s-s'') \mid t'' \leq t', s'' \leq s'\}.$$

As a result, under our limiting assumptions, we can model a wide portfolio of selections and transformations of features using this notation. Let us call τ_{fst} the specific operator for feature selection and transformation; τ_{fst} is defined column-by-column, using auxiliary transformations $\tau_{fst}^{i,j}$, where $1 \leq i \leq n$ and $\leq j \leq m$. For a specific component i, $P_{fst}[i]$ is a triple of numbers (T,S,F). The idea is that T represents the temporal displacement of the i-th attribute (whereas -1 represents de-selecting it), S represents the spatial displacement (in terms of semi-length of the edge of the outermost hypercube to be considered), and F represents the combining function. Thus, specific values of t and S entail considering a subset of values of A_i to be combined into a single value that substitutes the original one. Accordingly, for example, $P_{fst}[i] = (2,0,0)$ is interpreted as substituting the current value of A with the value of A two time units before the considered time. The semantics of other functions can be decided for specific applications. To ease the notation, if $P_{fst}[i] = (T,S,F)$, we let $T[i]$ (resp., $S[i]$, $F[i]$) denote the temporal (resp., spatial, functional) component; recall that $F[i]$ is a function of the type:

$$F[i] : 2^{dom(A)} \rightarrow dom(A)$$

Formally, let us define:

$$\tau_{fst}^{i,j}(\mathcal{I}, P) = F[i](\mathcal{A}(t-T[i], s-S[i])) \tag{3.2}$$

If we use the simplified notation $\tau_{fst}^{i,j}(P)$ when $\mathcal{I}$ is clear from the context, we can now define τ_{fst}; we shall do so by ignoring the columns with names, times, space, and classes, recalling that their information is

$$A_1 = \begin{bmatrix} 100 & 11 & 12 \\ 9 & 104 & 7 \\ 16 & 17 & 7 \\ 108 & 102 & 102 \end{bmatrix}, A_2 = \begin{bmatrix} 2 & 23 & 24 \\ 102 & 90 & 100 \\ 8 & 29 & 30 \\ 31 & 32 & 33 \end{bmatrix}, A_3 = \begin{bmatrix} 34 & 35 & 36 \\ 103 & 38 & 109 \\ 40 & 100 & 42 \\ 43 & 102 & 45 \end{bmatrix} \text{ at } t = 0$$

$$A_1 = \begin{bmatrix} 22 & 107 & 48 \\ 49 & 100 & 56 \\ 102 & 5 & 54 \\ 85 & 56 & 101 \end{bmatrix}, A_2 = \begin{bmatrix} 58 & 78 & 60 \\ 105 & 88 & 109 \\ 64 & 120 & 101 \\ 67 & 68 & 69 \end{bmatrix}, A_3 = \begin{bmatrix} 70 & 71 & 2 \\ 73 & 100 & 75 \\ 106 & 10 & 108 \\ 79 & 80 & 81 \end{bmatrix} \text{ at } t = 1$$

$$A_1 = \begin{bmatrix} 82 & 83 & 84 \\ 105 & 86 & 77 \\ 88 & 89 & 100 \\ 41 & 92 & 93 \end{bmatrix}, A_2 = \begin{bmatrix} 94 & 95 & 96 \\ 97 & 98 & 99 \\ 100 & 101 & 102 \\ 103 & 104 & 105 \end{bmatrix}, A_3 = \begin{bmatrix} 106 & 10 & 108 \\ 10 & 80 & 111 \\ 11 & 113 & 14 \\ 115 & 11 & 117 \end{bmatrix} \text{ at } t = 2$$

$$A_1 = \begin{bmatrix} 78 & 119 & 120 \\ 121 & 12 & 13 \\ 14 & 15 & 16 \\ 127 & 18 & 129 \end{bmatrix}, A_2 = \begin{bmatrix} 130 & 31 & 132 \\ 33 & 13 & 135 \\ 136 & 137 & 80 \\ 25 & 40 & 141 \end{bmatrix}, A_3 = \begin{bmatrix} 142 & 25 & 144 \\ 145 & 14 & 47 \\ 48 & 149 & 50 \\ 151 & 52 & 153 \end{bmatrix} \text{ at } t = 3$$

Figure 3.1: Twelve slices of an example data set $\mathcal{I}$.

implicitly used in the definition of $\tau_{fst}^{i,j}(P)$. Classes, in particular, have a very clear behavior, as they are unaffected by the preprocessing. Observe, however, that the number of resulting instances may decrease after pre-processing, due to the fact that there is not enough information to preprocess all instances; for example, if $T[i] = 1$ for any attribute A_i, then each name will have one less instance after the application of τ_{fst}. Thus, we have:

$$\tau_{fst}^{i}(\begin{bmatrix} N_1 & \dots & a_{1i} & \dots \\ \dots & \dots & \dots & \dots \\ N_1 & \dots & a_{m^*i} & \dots \\ \dots & \dots & \dots & \dots \\ N_z & \dots & a_{1i} & \dots \\ \dots & \dots & \dots & \dots \\ N_z & \dots & a_{m^*i} & \dots \end{bmatrix}, P) = \begin{cases} \begin{bmatrix} N_1 & \dots & \dots \\ \dots & \dots & \dots \\ N_1 & \dots & \dots \\ \dots & \dots & \dots \\ N_z & \dots & \dots \\ \dots & \dots & \dots \\ N_z & \dots & \dots \end{bmatrix} & \text{if } T[i] = -1 \\ \begin{bmatrix} N_1 & \dots & \tau_{fst}^{i,T[i]}(P) & \dots \\ \dots & \dots & \dots & \dots \\ N_1 & \dots & \tau_{fst}^{i,m^*-T[i]}(P) & \dots \\ \dots & \dots & \dots & \dots \\ N_z & \dots & \tau_{fst}^{i,T[i]}(P) & \dots \\ \dots & \dots & \dots & \dots \\ N_z & \dots & \tau_{fst}^{i,m^*-T[i]}(P) & \dots \end{bmatrix} & \text{otherwise} \end{cases} \tag{3.3}$$

In other words, applying a feature selection and transformation operator to a data set and a specific attribute entails either de-selecting that attribute, or transforming it. The most simple transformations are temporal, by which one obtains, for example, the lagged version of that attribute. More complicated transformations can be obtained by looking for the values of the same attributes at different (past) times and different (neighboring) locations and, in the most general cases, combining them using a function F.

$$A_1 = \begin{bmatrix} 31 \\ 52 \end{bmatrix}, A_2 = \begin{bmatrix} 98 \\ 101 \end{bmatrix}, A_3 = \begin{bmatrix} 80 \\ 113 \end{bmatrix} \text{ at } t = 2 \tag{3.5}$$

$$A_1 = \begin{bmatrix} 60 \\ 68 \end{bmatrix}, A_2 = \begin{bmatrix} 13 \\ 137 \end{bmatrix}, A_3 = \begin{bmatrix} 14 \\ 149 \end{bmatrix} \text{ at } t = 3 \tag{3.6}$$

Figure 3.2: The example data set of Figure 3.1, after applying a feature selection and transformation.

Functions are taken from a finite set, and can therefore be denoted by natural numbers as well. We let the code 0 denote the simplest substituting function, applicable when $\mathcal{A}(t - t', s - s')$ is a singleton; the remaining codes can be decided depending on the application. Finally, we obtain the whole preprocessing τ_{fst} by applying, in any order, the preprocessing $\tau^1_{fst}, \ldots, \tau^n_{fst}$. Thus:

$$\tau_{fst}(\mathcal{I}, P) = \tau^1_{fst}(\tau^2_{fst} \ldots (\tau^m_{fst}(\mathcal{I}, P)) \ldots) \tag{3.4}$$

In order to understand how feature selection and transformation work on a practical case, consider the following scenario. A data set $\mathcal{I}$ consists of 48 instances, each one contains the values of three attributes A_1, A_2, A_3 at one of 12 possible neighboring locations, at one of 4 time points t_0, t_1, t_2, t_3. We assume that the locations form a rectangular 3×4 shape. For convenience, visualize $\mathcal{I}$ at the four times separately, focusing on each attribute separately. The results are 12 *slices*, as in Figure 3.1. In this example, we assume to have a vocabulary of three combining functions, denoted by $\{0, 1, 2\}$. In particular, 0 represents the *uniform* combining function (i.e., all neighboring locations contribute the same), 1 represents the *linear* combining function (i.e., the contribution decreases in a linear way), and 2 represents the *hyperbolic* combining function (i.e., the contribution decreases in a non-linear way). We want to understand the effect of applying the following feature selection and transformation operator:

$$P_{fst} : [(2, 1, 0), (0, 0, 0), (0, 0, 0)]$$

In other words, the intended effect is that all three attributes are selected, no transformation is applied to A_2, A_3, and the following transformation is applied to A_1: each value is substituted by a uniform average of all values at the neighboring locations (distance 1) plus the value itself, two time points before. As a consequence, the resulting data set can only have information about time points 2 and 3, and, out of the 12 positions, only the two in the center. Thus, the resulting data set will have the aspect in Figure 3.2. An alternative way to see the same situation is shown in Figure 3.3, where different colors represent different time points, and the spatial locations are drawn to form a rectangular shape. As a final note, recall that, in real data sets, there may be several locations and several time points. Transformations have the effect of returning smaller data sets but, in the event of a real case, the information loss is usually negligible (unlike our example seems to suggest).

3.1.3 Semantics of Outlier Detection

Let us call τ_{od} the specific operator for outlier detection; τ_{od} is defined row-by-row. For a specific component j, $P_{od}[j] \in \{0, 1\}$, and we define τ_{od} by components; each component corresponds to an instance j, and it is teated by the operator τ^j_{od}. The idea is that the value in the j-th position represents the elimination, or not, of the j-th row. For example, $P_{od}[j] = 0$ (resp., 1) is interpreted as the j-th instance is not eliminated (resp., eliminated). Thus, we have:

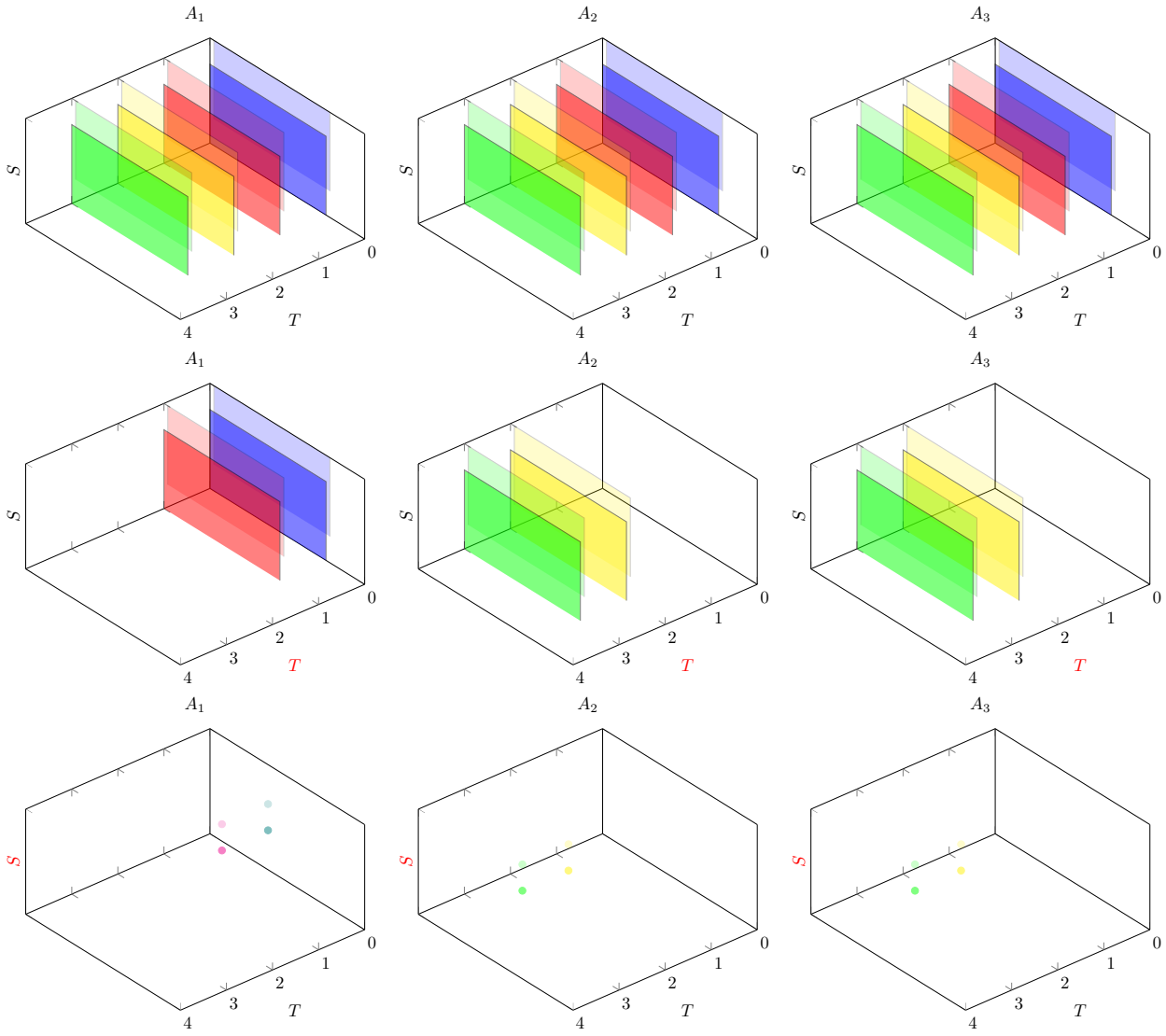


Figure 3.3: Example graphical representation.

$$\tau_{od}^{j}\left(\begin{bmatrix} N_1 & r_1 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_p & r_j & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_z & r_{m^*} & \dots & \dots \end{bmatrix}, P\right) = \begin{cases} \begin{bmatrix} N_1 & r_1 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_p & r_j & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_z & r_{m^*} & \dots & \dots \end{bmatrix} & \text{if } P_{od}[j] = 1 \\ \begin{bmatrix} N_1 & r_1 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_z & r_{m^*} & \dots & \dots \end{bmatrix} & \text{otherwise} \end{cases} \tag{3.7}$$

The entire preprocessing τ_{od} is obtained by applying, in any order, the preprocessing $\tau_{od}^1, \dots, \tau_{od}^m$:

$$\tau_{od}(\mathcal{I}, P) = \tau_{od}^1(\tau_{od}^2 \dots (\tau_{od}^m(\mathcal{I}, P)) \dots) \tag{3.8}$$

3.1.4 Semantics of Instance Selection

The case of instance selection behaves exactly as the previous one. Let us call τ_{is} the specific operator for outlier detection; again, τ_{is} is defined row-by-row. For a specific component j, $P_{is}[j] \in \{0, 1\}$, and we define τ_{is} by components, each component corresponds to an instance j, and it is treated by the operator τ_{is}^{j}. The idea is that the value in the j-th position represents the elimination, or not, of the j-th row. As such, for example, $P_{od}[j] = 1$ (resp., 0) is interpreted as the j-th instance is not eliminated (resp., eliminated). In other words, instance selection and outlier detection are opposite to each other. Thus, we have:

$$\tau_{is}^{j}(\begin{bmatrix} N_1 & r_1 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_p & r_j & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_z & r_{m^*} & \dots & \dots \end{bmatrix}, P) = \begin{cases} \begin{bmatrix} N_1 & r_1 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_p & r_j & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_z & r_{m^*} & \dots & \dots \end{bmatrix} & \text{if } P_{is}[j] = 0 \\ \begin{bmatrix} N_1 & r_1 & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ N_z & r_{m^*} & \dots & \dots \end{bmatrix} & \text{otherwise} \end{cases} \tag{3.9}$$

The whole preprocessing τ_{is} is obtained by applying, in any order, the preprocessing $\tau_{is}^{1}, \dots, \tau_{is}^{m}$:

$$\tau_{is}(\mathcal{I}, P) = \tau_{is}^{1}(\tau_{is}^{2} \dots (\tau_{is}^{m}(\mathcal{I}, P)) \dots) \tag{3.10}$$

The reason of having both outlier detection and instance selection, and to treat them separately, is that, generally, they are associated with different objectives and different evaluations. In short, outlier detection aims to identify the few instances that are the result of mistakes, or do not contribute to defining the problem. Their number is usually minimized and the evaluation is performed on the resulting data set. Instance selection aims to identify the subset of instances that, alone, represents the whole problem, avoiding duplicated instances or useless ones; their number is usually not included as objective.

3.2 NSGA-II: Multi-Objective Evolutionary Computation

3.2.1 EMO algorithm

As explained in Chapter 2, EMO algorithms (see Figure 3.4) consist of a main loop where there is an initial population (of solutions), whose members are evaluated and a new population is generated by using four main operations: *(i) selection*, *(ii) crossover*, *(iii) mutation*, and *(iv) elite-preservation.*This set of operations is performed until one or more stopping conditions are met. The final population constitutes a set of Pareto-optimal solutions. Then, EMO algorithms differ in selection, mutation, crossover and elite-preservation operations. We chose NGSA-II algorithm because it employs a strategy with a binary tournament selection and a better rank-crowding function, where the rank of an individual in a population is the non-domination level of the individual in the whole population.

3.2.2 Individual and Evaluation

The NSGA-II algorithm is one of the most used EMO procedures during the last decade. It is computationally fast, is available as open-source from the suite *jMetal* [94] and we can easily find research to compare it with. The *jMetal* package offers *simplicity* and ease of use, that is, the classes provided follow

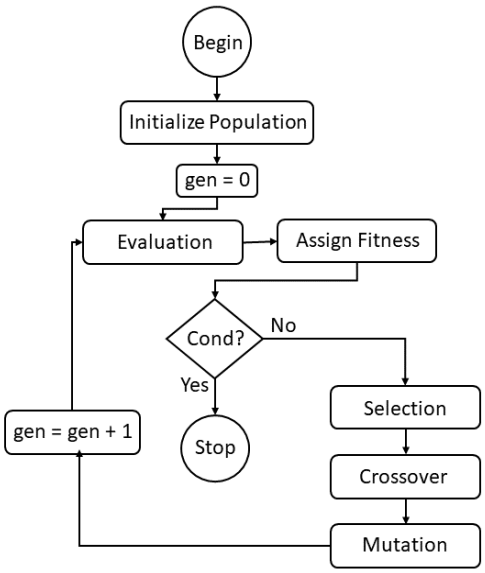


Figure 3.4: EMO algorithm scheme.

the principle that each component should only do one thing; *flexibility*, that is, the software incorporates a simple mechanism to execute the algorithms with different parameter settings; *portability*, that is, the framework and the algorithms developed with it run on machines with different architectures and/or with different operating systems; and *extensibility*, that is, new algorithms, operators and problems can be easily added to the framework. Therefore, we can easily use the semantics of P as representation of the individuals and define the objective functions and the crossover and mutation operators according to them. The entire operator P looks like:

$$P : [\underbrace{(T,S,F)_0,\ldots,(T,S,F)_{n-1}}_{P_{fst}} \mid \underbrace{B_n,\ldots,B_{(n+m)-1}}_{P_{od}} \mid \underbrace{B_{n+m},\ldots,B_{(n+2m)-1}}_{P_{is}}]$$

Each individual is represented as an operator P. To define the objective functions, as previously mentioned, we focused on the meta-problem rather than the problem itself, so, in addition to the performance of the learning algorithm, we are interested in measures related to data transformations that give us interpretability information. Then, objective functions generally are associated with the performance of the learning algorithm and another(s) associated to the dynamic preprocessing. As performance measures, they are state-of-the-art measures since the learning algorithm is a black-box. Some examples of performance measures are *correlation coefficient*, *accuracy*, *mean absolute error*, *specificity*, *sensibility*, *area under the ROC curve*, etc. Furthermore, different approaches have been used as an evaluation mode: *full training*, *k cross-validation*, *leave-one-out cross validation* or *test and training*. Then, as an objective function we have:

$$Eval(\mathcal{V}(\mathcal{I},P))$$

On the other hand, as measures related to data transformation we define the *cardinality* as the number of features of each solution. Cardinality is defined based on the P representation as follows:

$$Card_{fst}(P) = \sum_{i=1}^{n} \begin{cases} 0 & \text{if } P[i] = (-1,S,F) \\ 1 & \text{if } P[i] = (T,S,F), T \neq -1 \end{cases} \tag{3.11}$$

$$Card_{od}(P) = \sum_{i=n+1}^{n+m} \begin{cases} 0 & \text{if } P[i] = 1 \\ 1 & \text{if } P[i] = 0 \end{cases} \tag{3.12}$$

$$Card_{is}(P) = \sum_{i=n+m+1}^{n+2m} \begin{cases} 0 & \text{if } P[i] = 0 \\ 1 & \text{if } P[i] = 1 \end{cases} \tag{3.13}$$

Finally, these functions together with the previous one, can be seen as objective functions of the multi-objective optimization problem and equation 2.2 becomes:

$$\begin{cases} \min / \max Eval(\mathcal{V}(\mathcal{I}, P)), \\ \min / \max Card_q^1(P), \\ \dots \\ \min / \max Card_q^z(P) \end{cases} \tag{3.14}$$

where if $z = 1$ we have a two-objectives optimization problem; if $z = 3$ our problem is a three-objectives optimization problem; and if $z = 3$ we have a fourth-objective optimization problem.

3.2.3 Single-point Crossover Operator

We now have two solutions as two operator solutions. Depending on the preprocessing technique, the crossover is done in a determinated sequence of the solutions. As previously explained, each solution has three sequences: one of n triples and two of m pieces. Once the sequence to be worked on has been selected, the crossover operation consists of juxtaposing a certain part of the sequence of the first solution with another part of the sequence of the second solution, keeping the length of the operator. An example can be as follows: in the feature selection and transformations case, there should be P_{fst}^1, $P_{fst}^2 \in P$, as two sequences of n triples (T,S,F). Then, a randomly chosen integer $i \in \{1, \dots, n\}$ and generate $P_{fst}^{1'}$ by juxtaposing the first i triples from P_{fst}^1 with the last $n - i$ triples from P_{fst}^2; similarly for $P_{fst}^{2'}$ if needed. A pseudo code for this operation can be found in Algorithm 1.

3.2.4 Mutation Operator

We consider a solution seen as an operator solution. The mutation operation consists of flipping a component of the solution. Taking into account that our operator is made up of triples and pieces, the mutation will flip a complete triple or a piece, depending on the sequence we are working with. An example can be: in the outlier detection case, we are supposed to have $P_{od}^1 \in P_{od}$, represented as a sequence of m bits B. Then, we generate $P_{od}^{1'}$ by randomly flipping a bit from P_{od}^1. Flipping a bit consist of changing the B value with another B' value that is randomly generated. Bits' values can be mutated considering that the binary random values make sense in the solution. However, when mutating a triple, new values must be limited. For example, considering the T element, if we have data of 5 times before the current time, it makes no sense to generate a T value of 6. Now, we define two modes of flipping a triple: adding or subtracting 1 unit to the original value or changing the original value by randomly generating a new value in a limited range. Here, we defined the minimum range value as 0 for the three elements and as maximum range values: we called $maxT$ to the maximum value of the temporal element, $maxS$ to the maximum value of the spatial element and $maxF$ to the maximum value of the function element. A pseudo code of the mutation operation is shown in Algorithm 2.

Algorithm 1

procedure CROSSOVER
  $Input \leftarrow P1, P2, probability$
  $Output \leftarrow Offspring1, Offspring2$
  if $randomDouble < probability$ **then return** $(Offspring1, Offspring2) = (P1, P2)$
  else
    for $i \leftarrow 0 : (n + 2m) - 1$ **do**
      $internalProb \leftarrow randomInt(1, 3)$
      if $internalProb = 1$ **then**
        $crossPoint \leftarrow randomInt(0, n - 1)$.
        $Offspring1 \leftarrow juxtaposing$ triples of $P1[0 : crossPoint]$ with $P2[crossPoint : n - 1]$.
        $Offspring2 \leftarrow juxtaposing$ triples of $P2[0 : crossPoint]$ with $P1[crossPoint : n - 1]$.
      end if
      if $internalProb = 2$ **then**
        $crossPoint \leftarrow randomInt(n, (n + m) - 1)$.
        $Offspring1 \leftarrow juxtaposing$ bits of $P1[n : crossPoint]$ with $P2[crossPoint : (n + m) - 1]$.
        $Offspring2 \leftarrow juxtaposing$ bits of $P2[n : crossPoint]$ with $P1[crossPoint : (n + m) - 1]$.
      end if
      if $internalProb = 3$ **then**
        $crossPoint \leftarrow randomInt(n + m, (n + 2m) - 1)$.
        $Offspring1 \leftarrow juxtaposing$ bits of $P1[n+m : crossPoint]$ with $P2[crossPoint : (n+2m) - 1]$.
        $Offspring2 \leftarrow juxtaposing$ bits of $P2[n+m : crossPoint]$ with $P1[crossPoint : (n+2m) - 1]$.
      end if
    end for
  end if
end procedure

Algorithm 2

```
1: procedure MUTATION
2:     Input ← P, probability, maxT, maxS, maxF
3:     Output ← Offspring
4:     if randomDouble < probability then return Offspring = P
5:     else
6:         for i ← 0 : (n + 2m) − 1 do
7:             internalProb ← randomInt(1, 3)
8:             if internalProb = 1 then
9:                 mutPoint ← randomInt(0, n − 1).
10:                for j ← 0 : P(i).length do
11:                    if j = 0 then
12:                        if randomBinary = false then
13:                            Offspring ← T[mutPoint] = randomInt(0, maxT).
14:                        else
15:                            Offspring ← T[mutPoint] ± 1.
16:                        end if
17:                    end if
18:                    if j = 1 then
19:                        if randomBinary = false then
20:                            Offspring ← S[mutPoint] = randomInt(0, maxS).
21:                        else
22:                            Offspring ← S[mutPoint] ± 1.
23:                        end if
24:                    end if
25:                    if j = 2 then
26:                        if randomBinary = false then
27:                            Offspring ← F[mutPoint] = randomInt(0, maxF).
28:                        else
29:                            Offspring ← F[mutPoint] ± 1.
30:                        end if
31:                    end if
32:                end for
33:            end if
34:            if internalProb = 2 then
35:                mutPoint ← randomInt(n, (n + m) − 1).
36:                Offspring ← fliping bit in P[mutPoint].
37:            end if
38:            if internalProb = 3 then
39:                mutPoint ← randomInt(n + m, (n + 2m) − 1).
40:                Offspring ← fliping bit in P[mutPoint].
41:            end if
42:        end for
43:    end if
44: end procedure
```

Chapter 4

Application 1: Groundwater Aquifers

Within the last few decades, the continuous growth of the exploitation of water resources for industrial, human and agricultural purposes has brought great attention to the need for preserving the quality of groundwater [95, 96]. The complex reality of this sector has pushed the scientific community to research the proper management of water resources in order to improve knowledge and to protect every aspect of this resource. The goal of this research is to deal with the problems that originated from the variation of volume and intensity of precipitation due to climate change, over-exploitation, salinization, anthropic pollution, degradation, and large scale irrigation. An example of the need for a multidisciplinary approach is [97] but, in fact, many studies have demonstrated that a deliberate protection of existing water resources could contribute to the preservation of the availability of fresh water [98–100]. This chapter presents two of the problems concerning the preservation of the existing fresh water resources. Starting with a data set obtained from the physical-chemical analysis of several samples of groundwater in the same area, we focused our attention on two problems. First, we considered the presence of mercury in the samples, and we approached the problem of establishing a relationship between the physical-chemical spectrum of the sample and the amount of mercury. We used a simple regression to establish this relationship and then we used our dynamic preprocessing techniques to ensure the quality of the data, by searching for possible outliers. Therefore, our first problem is an *outlier detection for regression* problem. As such, individuals will be taking into account, as explained in the previous chapter, the possibility of eliminating instances in order to improve the performance of the regression. In this problem, we take advantage of our general setting that allows us to search for outliers in the context of a feature selection, with and without non-linear modifications of the data. Second, we approached the problem of establishing whether or not the groundwater has a specific physical-chemical fingerprint from which the specific aquifer (one of four possibilities) can be identified. We approached the problem by using the names of the aquifers as classes, and by interpreting it as a *feature selection for classification* problem. Once again, we take advantage of our general setting that allows us to try different non-linear contributions of the attributes, and different methods of defining the performances of a fingerprint. This chapter is based on [101–103].

4.1 Data and Statistical Analysis

We analyzed a data set of physical-chemical samples of underground water from a very specific area in northeastern Italy. These samples were collected as part of an ongoing investigation commissioned by the local Regional Agency for Environment and Prevention of the University of Ferrara. It consists of 229 samples extracted from 57 wells located in the province of Ferrara, all belonging to the most

superficial geological aquifer group, from 2010 to 2017. The hydro-stratigraphic units of interest were named from $A1$ to $A4$, from the most to the least superficial. Referring to our data as a matrix definition explained in Chapter 2, in this case, the sources are the aquifer groups which have four names, $A1$, $A2$, $A3$ and $A4$. Each sample contains 16 chemical-physical indicators that are our independent attributes and, for which there is no temporal or spatial information available. In summary, our data matrix consists of a matrix of 229 rows and 18 columns (N,$A_1, \ldots, A_{16}$). The independent attributes are Br (*Bromine*), Ca^{2+} (*Calcium*), Cl^- (*Chlorine*), Fe (*Iron*), HCO_3^- (*Bicarbonate*), I (*Iodine*), K^+ (*Potassium*), Mg^{2+} (*Magnesium*), NH_4^+ (*Ammonium cation*), NO_3^- (*Nitrate*), Na^+ (*Sodium*), SO_4^{2-} (*Sulfate*), Hg (*Mercury*), T (*Temperature*), Eh (*Reduction potential*), DO (*Chemical oxygen demand*), and $E.C.$ (*Electric conductivity*). Recalling that there are two problems, the first of which concerns outliers for regression, Hg and is the considered as the class (and therefore it will be numeric). However, in the one concerning aquifer fingerprints, Hg will be excluded and the name of the aquifer will be the (categorical) class. In this last case, Br, Eh, and DO will also be excluded. Prior to any learning phase, it is useful to understand the basic properties of the data. The analysis schema will be applied to every problem and will include the following steps:

- An assessment of the basic characteristics of the dataset;
- A normality study of each variable;
- A univariate correlation study among all variables.

The relevant statistical measures of the different independent attributes are shown in Table 6.1. These are: the (non-standardized) *mean*, the *p-value* associated with a Shapiro normality test (in which the null hypothesis is that the population is normally distributed), the *kurtosis*, and the *skewness* of each distribution. As indicated in the figures, none of the variables are normal (their p-values are well below 0.05), and they present very high levels of kurtosis and skewness, with Mg^{2+} and Ca^{2+} being the most evident examples. Figure 6.1 and Figure 6.2 show the graphical representation of the statistical behavior of each of the variables except for the temperature, which presented the closest-to-normal behavior and is, therefore, less informative. As indicated, the parameters show very erratic behavior, with the presence of a relevant percentage of outliers [104] and the fact that most variables do not display normal behavior can be considered as an argument against traditional statistical methods for fingerprint extraction. The correlation between elements can be seen in Table 4.2; the most evident ones are $E.C.$ with Cl^-, and $E.C.$ with HCO_3^-.

4.2 A Brief Background

Outlier detection (OD) has been approached with evolutionary algorithms in the literature. In [105], for example, OD is solved by harmony search and differential evolution, where a single objective is defined using the sparsity coefficient. Additionally, in [106], the anomaly detection problem is dealt with as a classification problem. Voronoi diagrams are used for the task of classification, and they are generated by a many-objective evolutionary computation where objectives are progressively added. They correspond to the classical performance metrics for classification and for Voronoi diagrams (accuracy, recall, number of cells, total empty volume, and so on). Finally, in [107], a multi-objective evolutionary algorithm is proposed to determine the best penalty factors of a sparse group and a lasso constraint is imposed on the learning objective of Adabost, which is combined with an autoencoder for image outlier detection. Applications of geochemical fingerprinting range in a wide set of contexts, from studies of ancient artifacts such as glass or ceramics [108], to mineral identification and the discovery of Jurassic-age kimberlite [109], to dust transport monitoring [110], to groundwater resources identification and this

feature	*min*	*max*	*mean*	*p-value*	*kurtosis*	*skewness*
Br	0.05	0.87	0.42	$0.10 * 10^{-2}$	0.11	0.23
DO	0.90	13.00	2.72	$1.70 * 10^{-24}$	13.51	2.90
Eh	-165.00	259.00	29.45	$4.77 * 10^{-6}$	-0.69	0.28
T	11.20	20.00	15.81	$1.16 * 10^{-18}$	5.37	0.55
$E.C.$	252.00	4175.00	1574.00	$1.36 * 10^{-18}$	2.45	1.08
HCO_3^-	140.00	1879.00	606.50	$3.54 * 10^{-11}$	4.44	1.18
Cl^-	0.50	1413.00	56.00	$2.43 * 10^{-21}$	2.81	1.25
SO_4^{2-}	0.50	143.00	18.05	$1.61 * 10^{-19}$	7.59	2.03
Ca^{2+}	26.00	18400.00	462.70	$1.91 * 10^{-29}$	64.02	7.03
Mg^{2+}	9.00	74740.00	381.33	$2.89 * 10^{-32}$	226.95	15.03
Na^+	8.00	763.30	183.52	$1.79 * 10^{-18}$	4.02	1.51
K^+	0.89	768.50	24.39	$1.27 * 10^{-29}$	42.85	6.13
NH_4^+	0.00	63062.00	2135.61	$2.01 * 10^{-27}$	43.06	5.51
Fe	0.00	41824.00	1501.52	$1.40 * 10^{-27}$	39.20	5.47
As	0.01	0.04	$3.23 * 10^{-3}$	$1.38 * 10^{-22}$	18.84	3.23
Hg	0.00	0.42	0.03	$2.72 * 10^{-30}$	49.97	5.49

Table 4.1: Some basic statistical measures of the variables.

	T	$E.C.$	HCO_3^-	Cl^-	SO_4^{2-}	Ca^{2+}	Mg^{2+}	Na^+	K^+	NH_4^+	Fe	As	Br	DO	Eh	Hg
T	1															
$E.C.$	0.050	1														
HCO_3^-	0.069	0.479	1													
Cl^-	0.026	0.956	0.249	1												
SO_4^{2-}	0.208	-0.329	-0.344	-0.294	1											
Ca^{2+}	0.002	0.189	0.109	0.159	-0.082	1										
Mg^{2+}	0.038	0.102	-0.019	0.132	-0.038	-0.010	1									
Na^+	-0.031	0.842	0.265	0.866	-0.364	-0.093	0.131	1								
K^+	0.012	0.248	0.049	0.248	-0.061	0.767	-0.009	-0.093	1							
NH_4^+	-0.005	0.366	0.345	0.305	-0.213	-0.096	-0.030	0.332	-0.096	1						
Fe	0.053	0.357	0.472	0.269	-0.141	-0.054	0.007	0.223	-0.062	0.112	1					
As	-0.003	-0.206	-0.073	-0.208	0.029	-0.051	-0.018	-0.154	-0.047	0.051	-0.004	1				
Br	0.041	0.723	-0.419	0.939	-0.599	0.703	0.765	-0.412	-0.057	-0.029	0.123	0.015	1			
DO	-0.223	-0.004	0.067	-0.094	0.287	-0.170	-0.206	0.237	-0.123	0.062	0.069	-0.013	-0.109	1		
Eh	-0.105	0.074	0.398	-0.128	-0.088	0.038	-0.010	0.092	0.027	-0.041	-0.441	0.063	-0.157	0.040	1	
Hg	-0.132	0.413	-0.063	0.456	-0.251	0.463	0.410	-0.254	-0.143	-0.038	0.430	-0.079	0.400	0.010	-0.114	1

Table 4.2: Correlation matrix.

study [111]. Groundwater resource analysis has been the focus of studies aimed at fingerprinting for different purposes. In [112], for example, the authors use fingerprinting to evaluate the occurrence of microorganic elements and to help with understanding the sources and the processes which may be controlling the transport and fate of emerging contaminants in the floodplain of the River Thames to the northwest of Oxford and in the River Lambourn, in South-East England. In [113], top and subsoil groundwater were sampled around a station in Tomiño, in North-East Spain, and analyzed to identify and quantify volatile fuel organic compounds as well as diesel range organic elements. Also, in [114], discriminant analysis was used to identify the most probable source of chloride salinity in groundwater samples based on their geochemical fingerprints. Finally, geochemical fingerprinting proved itself relevant for the study of the quality of food and beverages, especially wine, as shown in [115]. We consider that the fingerprint extraction problem can be seen as a feature selection problem, that is, the problem of establishing the *best* subset of chemical-physical parameters. Thus, we developed a model that consists of a feature selection with non linear transformation.

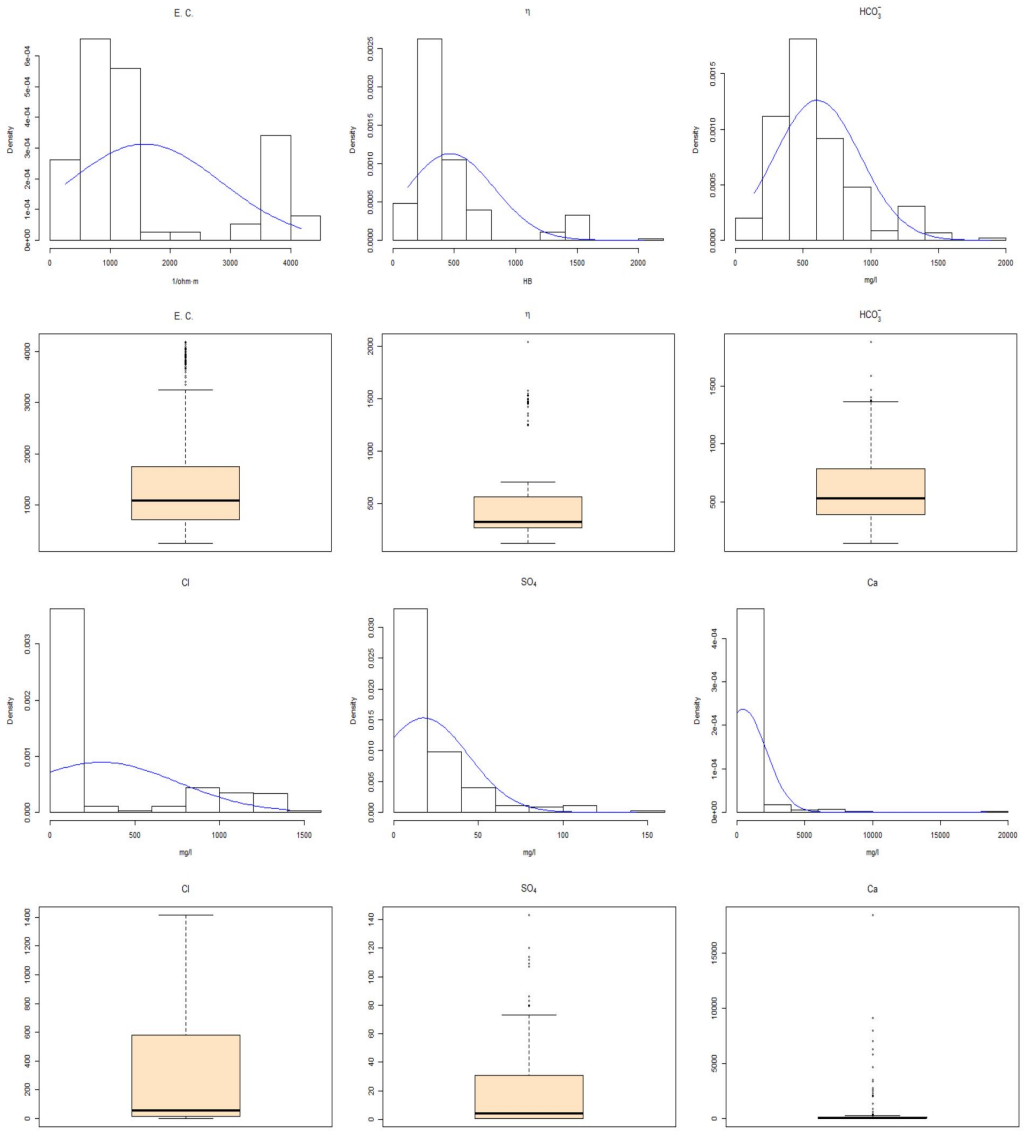


Figure 4.1: Distribution and outliers detection analysis: $E.C.$,η,HCO_3^-,Cl^-, SO_4^{2-},Ca^{2+}.

4.3 Learning Schemata for Outlier Detection

Recall that for this problem, we ignore the different names of the aquifers, and interpret the amount of mercury in each sample as the class. The problem to be solved involves identifying possible outliers in the regression model in which we link samples to the amount of mercury in them. We developed six models for outlier detection, all within our general schema as presented in the previous chapter. Model 1 consists of performing a feature selection (FS) alone, and model 2 consists of performing an OD alone. Now, considering that an instance could be an outlier only in the context of certain features, we applied the combined models of FS and OD and followed the sequential and simultaneous strategies. Thus, we have models 3 and 4 that consist of applying a FS+OD (resp. OD+FS) following the sequential strategy and model 5 that consists of performing a FS and OD simultaneously. Finally, considering that the variation in the amount of mercury can not be proportional to the variation of the chemical attributes, we developed model 6 that consists of applying a nonlinear feature transformation and OD simultaneously. The individual representations are as follows; in model 1 the individuals are represented as:

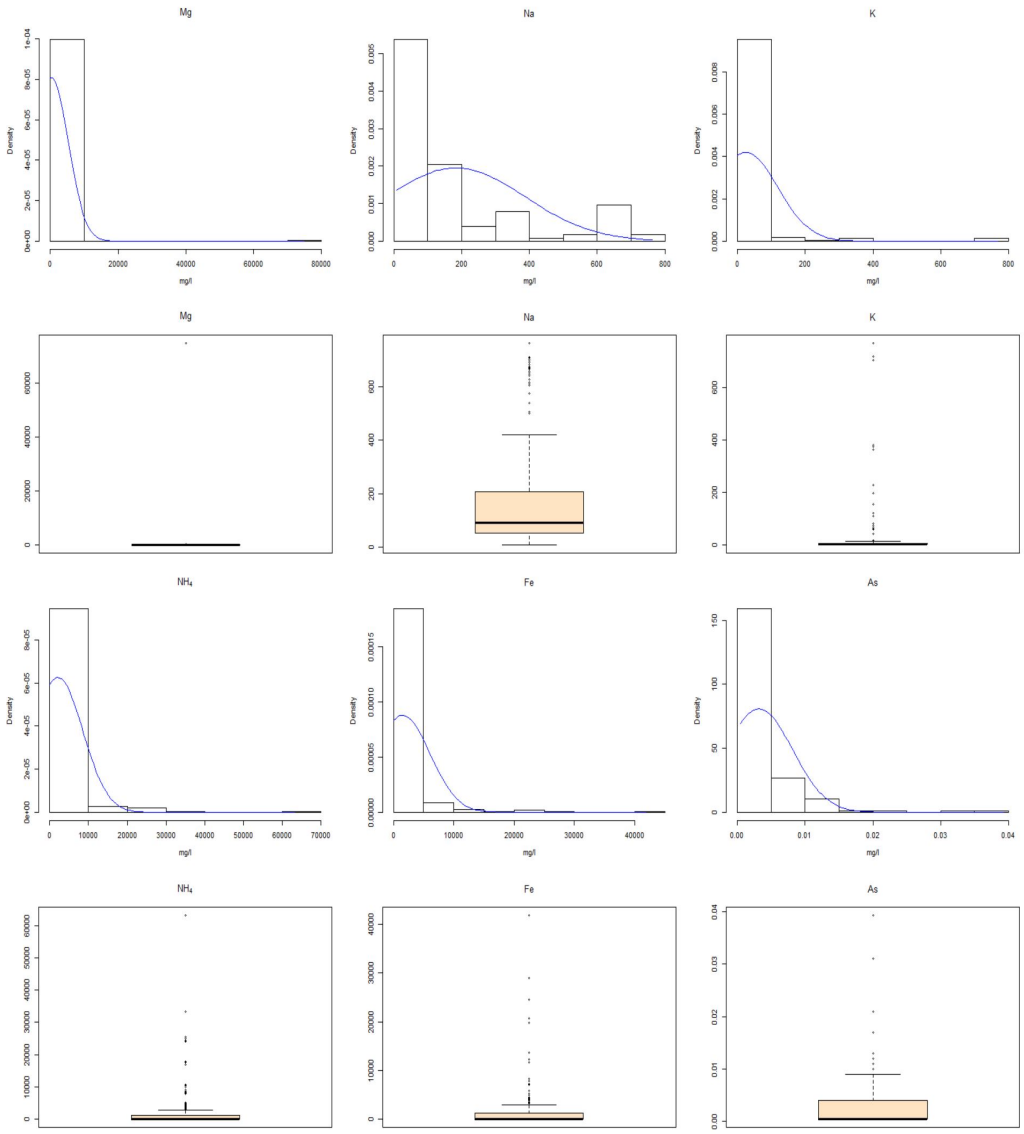


Figure 4.2: Distribution and outliers detection analysis: Mg^{2+},Na^{+},K^{+},NH_4^{+}, Fe,As.

$$P : \underbrace{[(T)_0, \ldots, (T)_{n-1}}_{P_{fst}}]$$

where $T[i]$ takes values of 0, that means that the attribute is selected; or -1, that means that the attribute is not selected. In model 2 the individual is represented as:

$$P : \underbrace{[B_n, \ldots, B_{m-1}}_{P_{od}}]$$

In model 3 (resp. model 4) when we apply FS (resp. OD) the individual is represented as in model 1 (resp. model 2); and when we sequentially apply OD (resp. FS) the individual is represented as in model 2 (resp. model 1). In model 5 the individual is represented as:

$$P : \underbrace{[(T)_0, \ldots, (T)_{n-1}}_{P_{fst}} \mid \underbrace{B_n, \ldots, B_{(n+m)-1}]}_{P_{od}}$$

where as in model 1, $T[i]$ takes values of 0, that means that the attribute is selected; or -1, that means that the attribute is not selected. Finally, in model 6 the individuals are represented as:

$$P : [\underbrace{(T,F)_0,\ldots,(T,F)_{n-1}}_{P_{fst}} \mid \underbrace{B_n,\ldots,B_{(n+m)-1}}_{P_{od}}]$$

where $T[i]$ takes values of 0, that means that the attribute is selected; or -1, that means that the attribute is not selected; and $F[i]$ takes values from 0 to the maximum function value $maxF$. As black box learning algorithm in all cases, we used the linear regression with the following default parameters: no embedded feature selection method, elimination of collinear features, and ridge penalty 10^{-8}, run in 10-fold *cross-validation* mode. Let $\mathcal{I}$ be our data set and the performance measure we used:

$$Corr(\mathcal{V}(\mathcal{I},P))$$

where $Corr$ measures the correlation between the stochastic variable obtained by the observations and the linear variable obtained by linear regression on the data set $\mathcal{I}$ after the transformation indicated by P. The correlation varies between -1 (perfect negative correlation) to 1 (perfect positive correlation), being 0 the value that represents no correlation at all. Furthermore, we used the cardinality, previously defined in Chapter 3, in all models as an interpretability measure. Thus, we have a two or three objective optimization equations depending on the model. Specifically, in model 1 we have a two-objective optimization equation:

$$\begin{cases} \max Corr(\mathcal{V}(\mathcal{I},P)), \\ \min Card_{fst}(P) \end{cases} \tag{4.1}$$

The same occurs in model 2, where the optimization equation is:

$$\begin{cases} \max Corr(\mathcal{V}(\mathcal{I},P)), \\ \min Card_{od}(P) \end{cases} \tag{4.2}$$

In the rest of models we have a three-objective optimization equation:

$$\begin{cases} \max Corr(\mathcal{V}(\mathcal{I},P)), \\ \min Card_{fst}(P), \\ \min Card_{od}(P) \end{cases} \tag{4.3}$$

4.4 Settings

Since NSGA-II does not allow for the managing of external constraints and since the solution space is particularly wide, we have hardwired a constraint on the number of selected outliers. Thereby, in models in which OD is performed, every solution with more than 10% outliers have been excluded. Also, solutions that emerge during the different generations are artificially given the worst possible values in each objective and are therefore discarded. In terms of parameterization, we proceeded as follows. In every experiment, the population size is 100 and after an initial preliminary test, we found that 10,000 total evaluations (100 generations) were sufficient to show convergent behavior. In models 3 and 4, each of the sequential phases were granted 10,000 evaluations, which means that the last population in a sequential experiment is the result of 20,000 evaluations. For this reason, in models 1, 2 and 5 the number of evaluations was set to 20,000. The maximum percentage of detected outliers was set to 10%. Also, we used the standard crossover and mutation, with probabilities (set after an initial explorative analysis) of 0.7 and 0.3, respectively. Finally, in terms of an *a posteriori* decision, in each experiment, we chose to select the individual with the best value of $Corr(\mathcal{V}(\mathcal{I},P))$. We ran 10 different executions for each model and the seeds for folder extraction are always fixed from 1 to 10, to ensure the repeatability of the experiments. In model 6, the maximum exponent was set to 3 and we have set each execution for a total of 10,000 evaluations.

ex	*cc*	*rmse*	*rae(%)*	*rrse(%)*
1	0.61	0.002	65.81	78.84
2	0.62	0.002	65.18	78.21
3	0.61	0.002	66.11	79.29
4	0.62	0.002	64.90	78.15
5	0.61	0.002	65.72	78.97
6	0.60	0.002	65.42	79.47
7	**0.63**	**0.002**	**64.47**	**77.70**
8	0.59	0.002	66.44	80.77
9	0.61	0.002	66.16	79.01
10	0.61	0.002	65.42	78.60
avg	0.611	0.002	65.564	78.901

Table 4.3: Evaluation of linear regression on each of the 10 data sets obtained by selecting the best individual from 10 executions; the seed was varied, and model 1 was used.

ex	*cc*	*rmse*	*rae(%)*	*rrse(%)*	*out(%)*
1	0.74	0.001	60.21	67.56	9.82
2	0.74	0.001	59.08	67.77	9.82
3	0.74	0.001	58.31	67.94	8.42
4	0.68	0.001	64.56	73.25	9.1
5	0.74	0.001	61.12	67.03	9.12
6	0.69	0.001	63.52	72.95	8.07
7	0.72	0.001	60.79	69.27	8.42
8	0.73	0.001	61.74	68.54	9.47
9	0.68	0.001	63.86	73.77	7.37
10	**0.76**	**0.001**	**59.05**	**65.01**	**3.86**
avg	0.720	0.001	61.222	69.309	8.350

Table 4.4: Evaluation of linear regression on each of the 10 data sets obtained by selecting the best individual from 10 executions; the seed was varied, and model 2 was used.

4.5 Results for Outlier Detection

The presence of features that are noise with respect to the problem is seen in Table 4.3, which shows the results of the experiments with the pure FS strategy. The following indicators are presented: *cc* (the correlation coefficient), *rmse* (the root mean squared error), *rae* (the relative absolute error), and *rrse* (the root relative squared error). When we conducted the OD experiments, it was noted that the correlation coefficients increased significantly. The last column of Table 4.4 indicates the rate of outliers that were selected and eliminated for the best individual in each experiment. As was seen, especially in some cases, the number was very low, which indicates that a correct choice of (suspected) outliers can truly improve the learning phase, even in the presence of noise features (recall that there was no feature selection). When we ran the simultaneous FS and OD experiment with model 5, we obtained the results in Table 4.5 which, as expected, proved that outliers should be detected relative to the features that are being used in the learning phase. In this table we included a final column # with the number of the selected features. While a simultaneous, integrated, and automated system is certainly preferable over a semiautomated system, such as a sequential strategy, for practical reasons, the question remains that if

ex	*cc*	*rmse*	*rae(%)*	*rrse(%)*	*out(%)*	#
1	0.73	0.002	62.08	68.34	7.96	14
2	0.74	0.001	61.27	67.14	5.62	15
3	0.73	0.001	59.18	68.15	9.13	14
4	0.73	0.001	58.17	68.71	8.43	13
5	0.71	0.002	62.39	69.67	9.84	11
6	0.72	0.001	58.94	69.13	8.43	13
7	0.74	0.001	58.81	67.46	8.89	15
8	0.73	0.001	58.79	67.91	6.56	14
9	0.71	0.001	60.88	70.57	6.79	15
10	**0.76**	**0.001**	**59.19**	**64.96**	**9.84**	**15**
avg	0.730	0.001	59.971	68.204	8.149	13.9

Table 4.5: Evaluation of linear regression on each of the 10 data sets obtained by selecting the best individual from 10 executions; the seed was varied, and model 5 was used.

ex	*cc*	*rmse*	*rae(%)*	*rrse(%)*	*out(%)*	#
1	**0.75**	**0.001**	**61.87**	**66.37**	**6.56**	**7**
2	0.71	0.002	63.71	70.47	8.66	6
3	0.69	0.002	64.80	71.45	6.56	6
4	0.70	0.002	63.61	71.34	7.96	7
5	0.68	0.002	63.25	73.39	9.37	6
6	0.69	0.002	63.33	72.11	7.73	5
7	0.72	0.001	62.95	68.68	8.20	7
8	0.73	0.002	61.90	68.21	8.66	6
9	0.74	0.001	60.79	67.35	9.84	8
10	0.68	0.002	63.49	73.41	7.49	9
avg	0.709	0.002	62.969	70.280	8.102	6.7

Table 4.6: Evaluation of linear regression on each of the 10 data sets obtained by selecting the best individual from 10 executions; the seed was varied, and model 3 was used.

selecting features and then detecting outliers, or the other way around, gives results that are comparable to those obtained with the simultaneous strategy. The results of two experiments that correspond to the two possible sequential strategies, shown in Table 4.6 and Table 4.7, respectively, prove that while both of them are comparable to the simultaneous strategy in terms of improvement over the original problem, the simultaneous strategy still presents a slight advantage over both sequential schemata. Results of experiments with model 6 are in Table 4.8, in which we show the correlation coefficient of the polynomial function, in 10-fold cross-validation, along with its *relative absolute error* and its *root relative squared error*. Results also show the number of features that correspond to the selection and the percentage of outliers that were detected (and eliminated). As observed, our optimization algorithm extracted non-trivial polynomial functions with a very high correlation coefficient and a small percentage of real outliers (as witnessed by the difference between correlation coefficient with and without their elimination). Table 4.9 summarizes the average results.

ex	*cc*	*rmse*	*rae(%)*	*rrse(%)*	*out(%)*	#
1	**0.77**	**0.001**	**59.38**	**63.84**	**7.73**	**13**
2	0.73	0.001	60.97	68.28	8.66	10
3	0.74	0.001	58.08	67.42	8.66	9
4	0.71	0.002	61.90	70.69	7.96	9
5	0.73	0.001	58.25	67.96	4.92	13
6	0.74	0.001	57.40	67.14	9.60	12
7	0.71	0.002	63.50	70.03	9.84	11
8	0.71	0.002	61.52	70.36	7.73	12
9	0.69	0.002	61.67	71.93	6.79	9
10	0.72	0.002	61.12	68.80	7.73	11
avg	0.72	0.001	60.38	68.64	7.96	10.9

Table 4.7: Evaluation of linear regression on each of the 10 data sets obtained by selecting the best individual from 10 executions; the seed was varied, and model 4 was used.

ex	*cc*	*rmse*	*rae(%)*	*rrse(%)*	*out(%)*	#
1	**0.80**	**0.001**	**57.58**	**60.47**	**9.13**	**10**
2	0.79	0.001	55.66	60.68	9.83	16
3	0.79	0.001	57.62	61.84	8.15	14
4	0.79	0.001	56.17	60.87	3.27	14
5	0.79	0.001	57.47	60.93	7.25	15
6	0.79	0.001	56.75	60.77	2.10	13
7	0.78	0.001	58.91	62.27	8.89	15
8	0.78	0.001	58.79	62.26	8.66	16
9	0.78	0.001	58.30	63.63	5.15	14
10	0.78	0.001	60.03	64.18	2.81	15
avg	0.79	0.001	57.73	61.79	6.52	14.2

Table 4.8: Evaluation of linear regression on each of the 10 data sets obtained by selecting the best individual from 10 executions; the seed was varied, and the model 6 was used.

model	*cc*	*rmse*	*rae(%)*	*rrse(%)*
1	0.61	0.002	65.56	78.90
2	0.72	0.001	61.22	69.31
3	0.73	0.001	59.97	68.20
4	0.71	0.002	62.97	70.28
5	0.73	0.001	60.38	68.64
6	0.79	0.001	57.73	61.79

Table 4.9: A summary of the results.

4.6 Learning Schemata for Fingerprinting

Recall that for this problem, we interpreted the names as classes in $\mathcal{I}$. So, $\mathcal{I}$ is labeled with four classes $A1, \ldots, A4$ that correspond to the four aquifers, and we then asked the question: what is the smallest and most accurate fingerprint that uniquely identifies an aquifer? In this case, the individuals are defined

as follows:

$$P : \underbrace{[(T,F)_0, \ldots, (T,F)_{n-1}]}_{P_{fst}}$$

where, as previously mentioned, since we had no temporal or spatial information, $T[i]$ takes values of 0 (that means that the i feature is selected), or -1 (that means that the i feature is not selected) and $F[i]$ takes values from 0 to the maximum function value, $maxF$. In this case, the classes are the aquifer names and we chose to model the fingerprint of an aquifer as the set of values that best represent an (ideal) sample of groundwater from that aquifer, that is, its centroid. Then, we denoted, by using $C_j(P)$, the centroid of the j-th aquifer ($1 \leq j \leq 4$, in our case) and computed precisely using the features that correspond to P. The performance measures of our model are defined as *simple accuracy* and *adjusted accuracy*. Indicated by $A(I)$ the (true) aquifer to which the instance I corresponds, we computed the *simple accuracy* of P over $\mathcal{I}$ as follows:

$$SimpleAcc(P) = \Sigma_{I \in \mathcal{I}} \begin{cases} 1 & \text{if } A(I) = argmin_{Aj}\{d(I, C_j(P))\} \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

where, to evaluate the distance between two instances $I = (a_1, \ldots, a_d)$ and $J = (a'_1, \ldots, a'_d)$, we used the well-known notion of *Euclidean distance*, as in (4.5), below:

$$d(I, J) = \sqrt{\Sigma_{i=1}^{d}(|a_i - a'_i|^2)} \tag{4.5}$$

Because of the particular nature of our problem, we can modify (4.4) to take into account that, although the existence of an impermeable layer between aquifers is hypothesized, infiltration may occur. Therefore, a misclassification can be graded as less severe if the expected aquifer confines with the true one. The *adjusted accuracy* takes this aspect into account:

$$AdjustedAcc(P) = \Sigma_{I \in \mathcal{I}} \begin{cases} 1 & \text{if } A(I) = argmin_{Aj}\{d(I, C_j(P))\} \\ \frac{1}{2} & \text{if } A(I) = argmin_{Aj}\{d(I, C_j(P))\} \pm 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

In addition, we considered the cardinality as an interpretability measure and finally, in order to take into account the nonlinear transformation, we defined two new measures, the *sum of all exponents*:

$$SumExp(P) = \Sigma_{i=1}^{n} F[i] \tag{4.7}$$

and the *maximum exponent*:

$$MaxExp(P) = \max_{i=1}^{n} F[i] \tag{4.8}$$

Now, all measures $SimpleAcc(P)$, $AdjustedAcc(P)$, $Card(P)$, $MaxExp(P)$ and $SumExp(P)$ can be combined to obtain four variants of the optimization equation (2.2):

$$\begin{cases} \max\ SimpleAcc(P), \\ \min\ SumExp(P), \\ \min\ Cardinality(P) \end{cases} \quad (1) \qquad \begin{cases} \max\ AdjustedAcc(P), \\ \min\ SumExp(P), \\ \min\ Cardinality(P) \end{cases} \quad (2)$$

$$\begin{cases} \max\ SimpleAcc(P), \\ \min\ MaxExp(P), \\ \min\ Cardinality(P) \end{cases} \quad (3) \qquad \begin{cases} \max\ AdjustedAcc(P), \\ \min\ MaxExp(P), \\ \min\ Cardinality(P) \end{cases} \quad (4)$$

Results of variants (1), (2), (3), and (4) will be tested and compared to each other in order to establish the best schema.

4.7 Settings

In this experiment, we used the standard parameters of the NSGA-II evolutionary algorithm, and implemented elementary variants of mutation and crossover for them to be specific to our solution format. To cope with the intrinsic unbalancing of our data (over 70% of the samples belong to $A1$), we operated a re-sampling, to obtain a training set with 10 samples per aquifer ($\mathcal{I}_{training}$), and left every other sample for testing ($\mathcal{I}_{test}$). The test was performed by applying the accuracy function(s) to $\mathcal{I}_{test}$ using the centroid and the selected attributes extracted from the chosen solution. For each of the four different multi-objective optimization variants, we executed 10 runs, each with a different seed. The population side was 100 in each experiment, and we set each experiment to run for 100 generations.

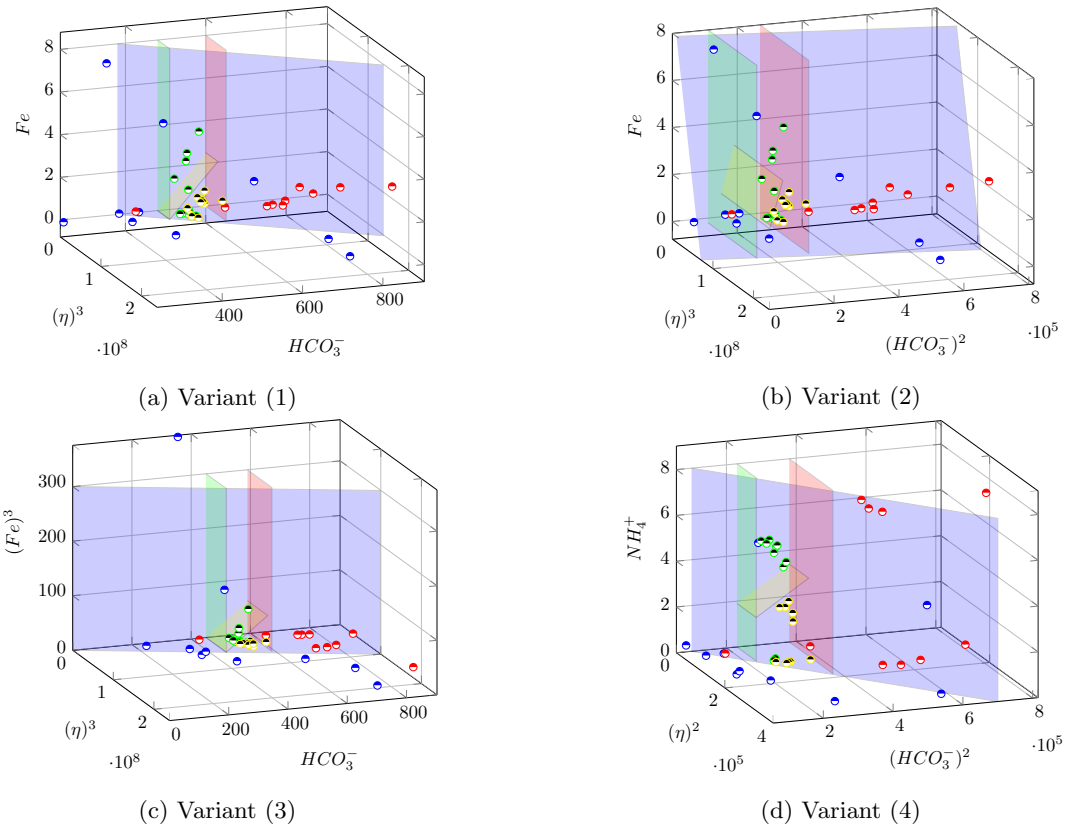
4.8 Results for Fingerprinting

For this next set of experiments, we used $\mathcal{I}_{training}$ and $\mathcal{I}_{test}$ without any further transformation. We obtained four sets of results, displayed in Table 4.10. The column *acc* indicates the test results for obtained accuracy, and the remaining columns show how this accuracy changes when analyzed by class, that is, how accurate is our model in identifying each of the four aquifers. As was observed, the general accuracy ranges from 0.54 to 0.66, which can be considered relatively high. By looking at the single class results, evidently, the aquifers $A3$ and $A4$ are identified with accuracies ranging from 0.9 to 1, and aquifer $A2$ is correctly identified with a rate from 0.71 to 0.81. Aquifer A_1 seemed to be the most difficult one. For each model, we identified the *most simple and accurate* solution with results that can be displayed, looking at the highest accuracies within fingerprints with less than four columns, and with a preference for lower exponents - these are indicated by a * in Table 4.10. In Figure 4.3, such solutions are displayed graphically, where their distinguishing power becomes evident. These results can be explained as follows. The Ca^{2+} and HCO_3^- levels are controlled by the interaction between rock and water, and related to the dissolution of carbonate and to the degradation of organic matter [116]. Hydrogen carbonate, in particular, is dissolved inorganic carbon in fresh waters, which is derived from the dissolution of calcite and dolomite, and its levels, therefore, implicitly express the concentrations of calcium and magnesium derived from these two minerals. Moreover, calcium and magnesium, together, define the level of hardness (η) of the water. Finally, iron and manganese, among others, are widely found in soils and aquifers, and have similar geochemical behavior. The reducing conditions, residence time, well depth and salinity are the key factors leading the dissolution and migration of Fe and Mn to groundwater [117]. This may explain our findings, that seem to indicate the HCO_3^-, Fe, and η, allows one to distinguish between the aquifers of the group under analysis. The fingerprinting problem can be also approached by looking into subsets of *ratios* among characteristics, instead of subsets of characteristics. In other words, instead of looking for the geochemical fingerprint of each aquifer among combinations of the chemical indicators, we examined different combinations of ratios of affine elements and quantities. This entailed pre-processing the data set to compute such ratios, and then applying the same optimization models. The reason behind this approach lies in the fact that the combinations of ratios of affine elements are sometimes better suited to identifying the geochemical signatures of an aquifer. This is because affine elements tend to be preserved during the dilution contribution of meteoric waters of reborn. Table 4.11 shows the ten executions with the four models. As indicated, some of the proposed solutions present very high accuracies; general accuracy now ranges from 0.79 to 0.91, and aquifer $A1$ is now correctly identified with accuracies from 0.75 to 0.91. For those fingerprints with three elements or less, their ability of discernment

			recall			
	fingerprint	*acc.*	*A1*	*A2*	*A3*	*A4*
variant (1)	η, $(HCO_3^-)^3$, $(NH_4^+)^2$, $(Fe)^2$	0.60	0.55	0.71	1.00	1.00
	η, $(HCO_3^-)^3$, $(NH_4^+)^2$, Fe	0.60	0.55	0.71	1.00	1.00
	$(\eta)^3$, $(HCO_3^-)^2$, $(Fe)^3$	0.60	0.55	0.71	0.90	1.00
	$(T)^2$, η, $(HCO_3^-)^3$	0.55	0.49	0.81	0.90	0.75
	η, $(HCO_3^-)^2$, $(Na^+)^2$, NH_4^+	0.54	0.47	0.81	1.00	1.00
	$(\eta)^3$, HCO_3^-, Fe	0.60	0.55	0.71	0.90	1.00
	$(\eta)^3$, $(HCO_3^-)^2$, $(Fe)^3$	0.60	0.55	0.71	0.90	1.00
	$(T)^2$, η, HCO_3^-	0.55	0.49	0.81	0.90	0.75
	η, $(HCO_3^-)^3$, $(Na^+)^3$, $(Fe)^3$	0.54	0.47	0.71	1.00	1.00
*	$(\eta)^3$, HCO_3^-, Fe	0.60	0.55	0.71	0.90	1.00
variant (2)	$(\eta)^2$, $(HCO_3^-)^3$, $(NH_4^+)^3$, $(Fe)^2$	0.66	0.55	0.71	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, $(Fe)^3$, $(As)^2$	0.66	0.55	0.71	0.90	1.00
	$(\eta)^2$, HCO_3^-, $(NH_4^+)^2$	0.64	0.53	0.81	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, NH_4^+, $(Fe)^2$	0.66	0.55	0.71	1.00	1.00
	η, $(HCO_3^-)^2$, $(Na^+)^2$, $(NH_4^+)^3$, $(As)^2$	0.62	0.47	0.81	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, NH_4^+, Fe	0.66	0.55	0.71	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, $(NH_4^+)^3$, $(Fe)^2$	0.66	0.55	0.71	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, NH_4^+, $(Fe)^3$	0.66	0.55	0.71	1.00	1.00
*	$(\eta)^3$, $(HCO_3^-)^2$, Fe	0.66	0.55	0.71	0.90	1.00
	η, $(HCO_3^-)^2$, $(Na^+)^2$, $(NH_4^+)^2$, $(Fe)^3$	0.62	0.47	0.81	1.00	1.00
variant (3)	$(\eta)^2$,HCO_3^-, $(NH_4^+)^3$	0.59	0.53	0.48	1.0	1.00
	$(\eta)^3$, HCO_3^-, $(Fe)^3$	0.60	0.55	0.43	0.90	1.00
	$(\eta)^3$, $(HCO_3^-)^3$, $(Fe)^3$	0.60	0.55	0.43	0.90	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, $(NH_4^+)^2$, Fe	0.60	0.55	0.48	0.90	1.00
	$(T)^2$, $(HCO_3^-)^2$, SO_4^{2-}, NH_4^+	0.54	0.48	0.43	1.00	0.75
	$(\eta)^3$, $(HCO_3^-)^3$, $(Fe)^3$	0.60	0.55	0.43	0.90	1.00
	$(\eta)^3$, $(HCO_3^-)^3$, $(Fe)^3$	0.60	0.55	0.43	0.90	1.00
	$(T)^2$, η, $(HCO_3^-)^2$	0.55	0.49	0.43	1.00	0.75
*	$(\eta)^3$, HCO_3^-, $(Fe)^3$	0.60	0.55	0.43	0.90	1.00
	$(T)^2$, η, HCO_3^-	0.55	0.49	0.43	1.00	0.75
variant (4)	$(\eta)^2$, $(HCO_3^-)^3$, $(NH_4^+)^3$, $(Fe)^2$	0.66	0.55	0.71	1.00	1.00
	η, HCO_3^-, NH_4^+, Fe	0.64	0.54	0.81	0.90	1.00
*	$(\eta)^2$, $(HCO_3^-)^2$, NH_4^+	0.64	0.53	0.81	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, $(NH_4^+)^2$, NH_4^+, $(As)^2$	0.66	0.55	0.71	1.00	1.00
	η, $(HCO_3^-)^2$, $(Na^+)^2$, $(NH_4^+)^3$, $(As)^2$	0.62	0.47	0.81	1.00	1.00
	η, $(HCO_3^-)^2$, $(Na^+)^2$, $(NH_4^+)^3$	0.62	0.47	0.81	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^2$, $(NH_4^+)^2$	0.64	0.53	0.81	1.00	1.00
	η, $(HCO_3^-)^2$, Na^+, $(NH_4^+)^2$, $(Fe)^2$, $(As)^3$	0.62	0.47	0.81	1.00	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, $(NH_4^+)^2$, $(Fe)^3$	0.66	0.55	0.71	1.00	1.00
	η, $(HCO_3^-)^2$, $(Na^+)^2$, NH_4^+, $(Fe)^2$	0.62	0.47	0.81	1.00	1.00

Table 4.10: Results of the characteristics-based fingerprinting experiment. From top to bottom: variants (1), (2), (3), and (4). Starred items indicate the best results of each variant.)

can be also shown graphically (see Figure 4.4); thus, the most simple and accurate solution for each model is shown. As can be seen, using ratios increases the accuracy of our characterization in a practical manner; the ratios that emerged as those with a better discernment ability can be explained as follows. First, electrical conductivity has been used to characterize groundwater by several authors [118, 119], the presence of inorganic suspended solids such as chloride, nitrate, phosphate, and sulfate ions (ions that



(a) Variant (1)

(b) Variant (2)

(c) Variant (3)

(d) Variant (4)

Figure 4.3: Graphical representation of the starred solutions from Table 4.10. Figure (a): η^3, HCO_3^-, Fe. Figure (b): $\eta^3, {HCO_3^-}^2, Fe$. Figure (c): η^3, HCO_3^-, Fe^3. Figure (d): $\eta^2, {HCO_3^-}^2, NH_4^+$.

carry a negative charge), or aluminum, calcium, magnesium, iron, and sodium ions (ions that carry a positive charge) may affect electrical conductivity. Electrical conductivity values may reflect both the recharge dynamics and the possible excessive pumping of wells. Second, the concentration ratios between alkaline earth elements could be correlated to water-sediment interaction times, and the contribution of fossil water content trapped in deep sediments [120]. In conclusion, the fingerprints that emerge from our analysis seem to indicate that the different aquifers can be distinguished from these elements: freshwater/fossil water mixing and saltwater contamination.
After an analysis of the results, it is evident that the following pair of ratios:

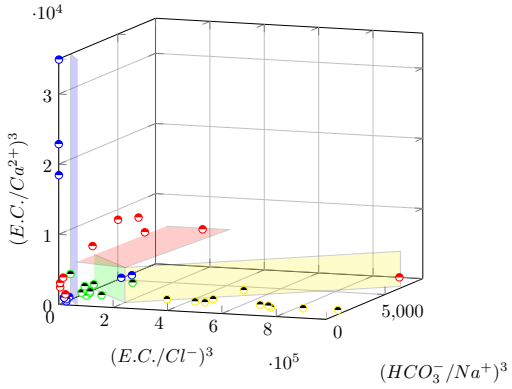
$$E.C./Cl^-, E.C./Ca^{2+}$$

is able to distinguish the four aquifers with the most accuracy. By reading the linear boundaries between values, it is noticeable that each aquifer is characterized by the following intervals:

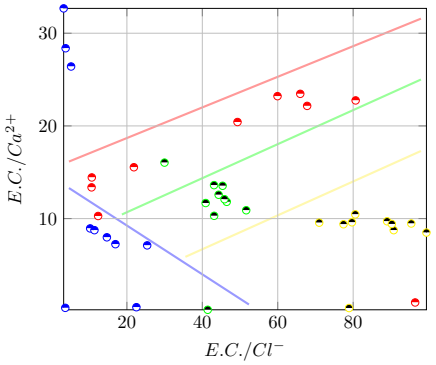
$$\begin{aligned}
A1: &\quad \text{under} && E.C./Ca^{2+} = -0.272 \cdot E.C./Cl^- + 14.251 \\
A2: &\quad \text{between} && E.C./Ca^{2+} = 0.16 \cdot E.C./Cl^- + 7.267 \text{ and} \\
& && E.C./Ca^{2+} = 0.1748 \cdot E.C./Cl^- + 15.475 \\
A3: &\quad \text{between} && E.C./Ca^{2+} = 0.161 \cdot E.C./Cl^- + 0.049 \text{ and} \\
& && E.C./Ca^{2+} = 0.16 \cdot E.C./Cl^- + 7.267 \\
A4: &\quad \text{under} && E.C./Ca^{2+} = 0.161 \cdot E.C./Cl^- + 0.049
\end{aligned}$$

			recall			
	fingerprint	*acc.*	*A*1	*A*2	*A*3	*A*4
variant (1)	$E.C./Cl^-$, HCO_3^-/Ca^{2+}	0.84	0.90	0.52	0.50	1.00
	$(E.C./Cl^-)^3$, $(\eta/Cl^-)^2$, $(Na^+/T)^2$, $(HCO_3^-/Ca^{2+})^2$	0.86	0.90	0.52	0.80	1.00
*	$(E.C./Cl^-)^3$, $(HCO_3^-/Na^+)^3$, $(E.C./Ca^{2+})^3$	0.85	0.90	0.48	0.80	1.00
	$(\eta/Cl^-)^3$, HCO_3^-/Ca^{2+}	0.79	0.81	0.62	0.90	1.00
	$E.C./Cl^-$, η/Cl^-, $(\eta/T)^3$	0.85	0.91	0.43	0.80	1.00
	$(Cl^-/Na^+)^2$, $E.C./Cl^-$, Na^+/T, $(HCO_3^-/\eta)^2$	0.83	0.90	0.43	0.50	1.00
	$(E.C./Cl^-)^3$, $(HCO_3^-/T)^2$	0.79	0.79	0.67	0.90	1.00
	$(E.C./Cl^-)^2$, $(\eta/Cl^-)^2$	0.85	0.90	0.48	0.80	1.00
	$(E.C./Cl^-)^2$, Ca^{2+}/K^+, $(E.C./Ca^{2+})^2$	0.75	0.82	0.24	0.70	1.00
	$(E.C./Cl^-)^3$, $(HCO_3^-/T)^3$, $(\eta/T)^3$	0.84	0.86	0.62	0.90	1.00
variant (2)	η/Cl^-, $(\eta/Na^+)^3$	0.90	0.82	0.43	0.80	1.00
	$(NH_4^+)^2$, $(E.C./Cl^-)^3$, $(\eta/Cl^-)^3$	0.91	0.90	0.48	0.70	1.00
	$(\eta/Cl^-)^2$, $E.C./Ca^{2+}$	0.88	0.75	0.57	0.90	1.00
	$(E.C./Cl^-)^3$, $(\eta/Cl^-)^3$, Na^+/T	0.87	0.90	0.43	0.80	1.00
	$(E.C./Cl^-)^3$, $(\eta/Cl^-)^2$,$(\eta/Na^+)^3$	0.91	0.90	0.48	0.80	1.00
	$(E.C./Cl^-)^3$, Na^+/T, η/T	0.89	0.76	0.67	1.00	1.00
	$(E.C./Cl^-)^2$, $(HCO_3^-/\eta)^3$, $(HCO_3^-/T)^3$	0.86	0.79	0.67	0.90	1.00
	Cl^-/Na^+, $E.C./Cl^-$, Na^+/T	0.91	0.90	0.43	0.50	1.00
	$E.C./Cl^-$, η/Cl^-, $(\eta/T)^3$	0.90	0.91	0.43	0.80	1.00
*	$E.C./Cl^-$, $E.C./Ca^{2+}$	0.91	0.90	0.48	0.60	1.00
variant (3)	$(Cl^-/Na^+)^2$, $(E.C./Cl^-)^2$, $(\eta/Cl^-)^2$	0.85	0.90	0.48	0.80	1.00
*	$E.C./Cl^-$, $(\eta/Cl^-)^2$	0.85	0.90	0.48	0.80	1.00
	NH_4^+, $E.C./Cl^-$, η/T	0.83	0.90	0.38	0.50	1.00
	$(NH_4^+)^3$, $(\eta/Cl^-)^2$, $(HCO_3^-/Ca^{2+})^2$, $E.C./\eta$	0.81	0.83	0.67	0.80	1.00
	T, $(\eta/Cl^-)^2$, HCO_3^-/Ca^{2+}	0.79	0.80	0.62	0.90	1.00
	η/Cl^-, HCO_3^-/Ca^{2+}	0.79	0.81	0.62	0.90	1.00
	As, $E.C./Cl^-$, $E.C./Na^+$	0.84	0.91	0.48	0.50	1.00
	$(E.C./Cl^-)^2$, $(Na^+/T)^2$	0.83	0.90	0.43	0.50	1.00
	η/Cl^-, $E.C./Ca^{2+}$	0.75	0.75	0.57	0.90	1.00
	η/Cl^-, HCO_3^-/T	0.66	0.60	0.81	1.00	1.00
variant (4)	$(E.C./Cl^-)^2$, $(E.C./Ca^{2+})^2$	0.90	0.90	0.48	0.60	1.00
	$E.C./Cl^-$, η/Cl^-, HCO_3^-/Na^+	0.91	0.90	0.48	0.80	1.00
	$(NH_4^+)^2$, $(\eta/Cl^-)^3$, $(HCO_3^-/Ca^{2+})^3$, $(E.C./HCO_3^-)^3$	0.88	0.82	0.67	0.80	1.00
	$(\eta/Cl^-)^3$, $(HCO_3^-/Ca^{2+})^2$, $(T/K^+)^2$	0.87	0.81	0.62	1.00	1.00
	$E.C./Cl^-$, η/Cl^-	0.91	0.90	0.48	0.80	1.00
	$(E.C./Cl^-)^3$, $(\eta/Cl^-)^2$	0.91	0.90	0.48	0.80	1.00
	$(\eta)^2$, $(HCO_3^-)^3$, η/Cl^-, $(HCO_3^-/Ca^{2+})^3$	0.89	0.88	0.57	0.90	1.00
	η/Cl^-, $(HCO_3^-/Ca^{2+})^3$, η/T	0.86	0.78	0.62	0.90	1.00
	$E.C./Cl^-$, $(\eta/Cl^-)^2$, η/T	0.91	0.91	0.43	0.80	1.00
*	$E.C./Cl^-$, $E.C./Ca^{2+}$	0.90	0.90	0.48	0.60	1.00

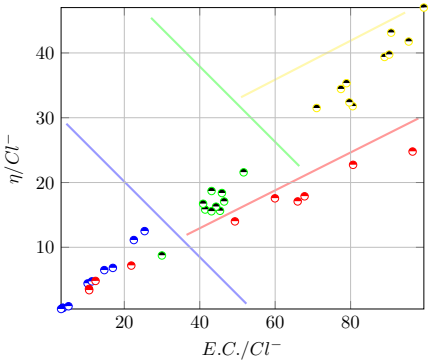
Table 4.11: Results of the ratios-based fingerprinting experiment. From top to bottom: variants (1), (2), (3), and (4). Starred items indicate the best results for each variant.



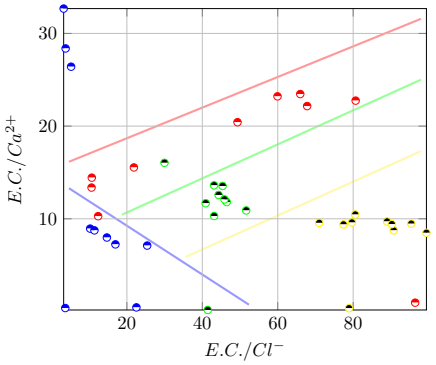
(a) Variant (1)



(b) Variant (2)



(c) Variant (3)



(d) Variant (4)

Figure 4.4: Graphical representation of the starred solutions from Table 4.11. Figure (a): $(E.C./Ca^{2+})^3, (E.C./Cl^-)^3$. Figure (b): $(E.C./Ca^{2+}), (E.C./Cl^-)$. Figure (c): $(\eta/Cl^-), (E.C./Cl^-)$. Figure (d): $(E.C./Ca^{2+}), (E.C./Cl^-)$.

Chapter 5

Application 2: Air Pollution Monitoring Stations

Anthropogenic environmental pollution is a known and indisputable issue. In everyday life, the human body is exposed to harmful substances in various ways, including the consumption of food and drink or via absorption with breathing. While it is possible to limit the absorption of harmful chemical compounds with nourishment by selecting the right products, it is impossible to choose the air you breathe. Every day, with every breath, chemicals that are potentially dangerous to health become absorbed into the lungs and body, particularly if they are suspended particulate matters, nitrogen oxides, CO, and SO_2. Exposure to each of these pollutants adversely affects human health [121]. This is confirmed by numerous scientific studies conducted over many years, around the world: Europe [122], and, in particular, Poland [123,124], Deutschland [125], and Italy [126], USA [127–129], Canada [130], Australia [131], Chile [132], among many others. Air quality is regularly monitored and alert systems inform residents about the forecasted or ensuing high concentrations of air pollutants. Undoubtedly, actions leading to the reduction of air pollution are definitely more effective. Generally, anthropogenic sources of air pollution are the common. Due to the development of civilization, it is impossible to completely eliminate them. However, it is possible to monitor their behaviour, and predict their effect in the short and medium-term. This chapter considers an air pollution prediction problem and, in particular, the problem of establishing how nitrogen oxides are correlated to the amount of traffic and to the weather conditions at a specific geographical point, in a crossroad in Wrocław, Poland. We define this problem within our framework as a *feature selection and transformation* problem for regression. This chapter is based on [133–137].

5.1 Data

There is only one communication station for measuring the air quality in the city of Wrocław, and it is located within a wide street with two lanes in each direction (GPS coordinates: 51.086390 North, 17.012076 East). The center of one of the largest intersections in the city with 14 traffic lanes is located approximately 30 meters from the measuring station; this area is observed by a traffic monitoring system. The measurement station is located on the outskirts of the city, at $9.6kms$ from the airport. Pollution data are collected by the Provincial Environment Protection Inspectorate and encompasses the hourly NO_2 and NO_x concentration values during three full years, from 2015 to 2017. The traffic data (denoted by t, in our results) are provided by the Traffic Public Transport Management Department of the Roads and City Maintenance Board in Wrocław, and include hourly count of all types of vehicles passing the intersection. Public meteorological data are provided by the Institute of Meteorology and Water Management, and they include: *air temperature* (a), *solar duration* (d), *wind speed* (w), *relative humidity*

feature	*min*	*max*	*mean*	*p-value*	*kurtosis*	*skewness*
a	-15.7	37.7	10.9	$7.21 * 10^{-31}$	-0.412	0.200
d	0	1	0.23	-	-0.113	1.287
w	0	19	3.13	$1.65 * 10^{-42}$	1.614	1.065
r	20	100	74.9	$2.82 * 10^{-35}$	-0.453	-0.698
p	906	1028	1003	$2.87 * 10^{-36}$	0.975	-0.359
t	30	6713	2771	$4.9 * 10^{-49}$	-1.567	-0.175
NO_2	1.7	231.6	50.4	$9.03 * 10*^{-26}$	1.786	0.767
NO_x	3.9	1728.0	142.2	-	16.302	2.553

Table 5.1: Descriptive statistics.

	a	d	w	r	p	t
a	1					
d	0.461	1				
w	0.0121	0.146	1			
r	-0.561	-0.641	-0.269	1		
p	-0.178	0.039	-0.211	0.044	1	
t	0.247	0.403	0.198	-0.439	-0.034	1
NO_2	0.140	0.081	-0.249	-0.189	-0.439	0.045
NO_x	-0.126	0.010	-0.214	0.067	0.104	0.444

Table 5.2: Correlation matrix.

(h), and *air pressure* (p). For the sake of uniformity, solar duration values have been re-normalized in the real interval $[0, 1]$ (as standard). The full data set contains 26,304 observations. In the preprocessing phase, the instances with at least one missing value (617 samples, 2.3%) were excluded. In summary, the data matrix consists of a matrix with one named attribute, the Station name (since it is the same name for all instances, we do not take into account this attribute); 26,304 rows; 6 independent attributes a, d, w, h, p and t; one time T attribute in which temporal information is codified hourly; and one class attribute that takes values of NO_2 or NO_x.

5.2 Statistical Analysis

The relevant statistical measures of the attributes are shown in Table 5.1: the (non-standardized) *mean*, the *p-value* associated with a Shapiro normality test (in which the null hypothesis is that the population is normally distributed), the *kurtosis*, and the *skewness* of each distribution. As was observed, variables are no-normal (their p-values are considerably below 0.05), and they present with lower levels of kurtosis and skewness, being NO_x and w are the examples with higher levels. In Figure 5.1 and 5.2 we included the graphical representation of the statistical behaviour of each of the variables. As noted, the parameters demonstrated erratic behaviour, with the presence of a relevant percentage of outliers and most variables did not show a normal behaviour. The correlation between elements can be seen in Table 5.2; the most evident ones are t with d, and t with NO_x.

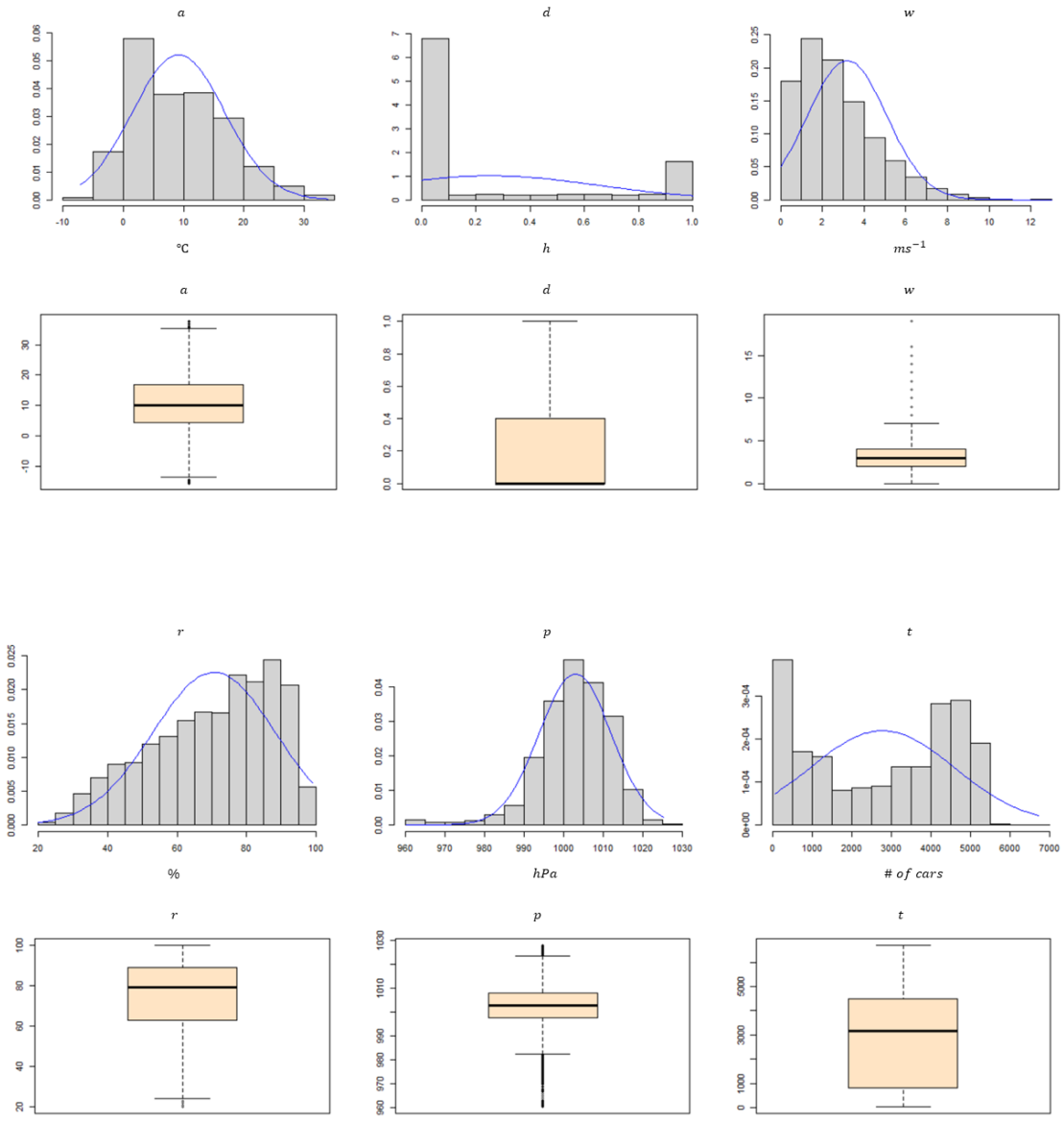


Figure 5.1: Distribution and outliers detection analysis: a, d, w, h, p, t.

5.3 Learning Schemata

The first environmental study that relates air pollution and meteorological variables and traffic conditions in Wrocław (Poland) was presented in [138]. The overall goal of the study was to determine how the levels of specific pollutants, namely, NO_2, NO_X, and $PM_{2.5}$, are related to the values of other attributes, such as weather conditions and traffic intensity, with the purpose of building an explanation model. In it, the value of a pollutant at a certain time instant is linked to the value of the predictor attributes from the same time instant. In [138], a non-linear, non-interpretable, atemporal model was used. The fitting ability of a non-interpretable model partially compensates for not using the historical values of the predictor, giving rise to a relatively good explanation model. In [139], the authors use random forests in an attempt to model the regression relationships between concentrations of NO_2, NO_x and $PM_{2.5}$, and nine variables describing meteorological conditions, temporal conditions and traffic flow. Furthermore, in the literature, there are mathematical models that predict concentrations of pollutants and recognize the

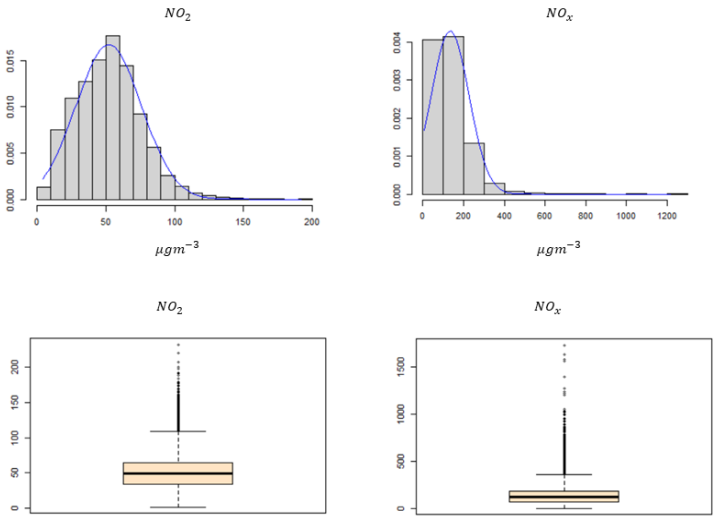


Figure 5.2: Distribution and outliers detection analysis: NO_2 and NO_x.

factors that have the greatest impact on them; these are called *autoregressive models* [140, 141]. On the other hand, *land use regression models* (*LUR*s) are a cost-effective approach for predicting the variability within ambient air pollutant concentrations with high resolution, shortening the time of exposure to the contaminants' harmful effects and reducing the intensity of this action; examples include [142, 143] (see also [144] for a survey). These models are often developed based on a series of previous observations however, it is unclear how well such models perform when extrapolated over time [145]. Moreover, in addition to the identification of the factor itself, the time of its operation also plays a significant role. For example, a momentary gust of wind results in a much smaller evacuation of pollutants and, thus, a decrease in concentration, when compared with the wind with the same speed persisting for several hours. To better explain a contamination concentration and taking the temporal component into account, one may want to identify not only the factors at play but also the moment in time for which their influence is the greatest, and build a *temporal land use regression model* (*TLUR*). By taking into account the temporal component, a TLUR is more accurate and more interpretable than a LUR. In order to identify the factors that most influence the amount of air pollutant and the time of its operations, we looked at the temporal information of our data and we developed three models. Model 1 consists of a lag selection only; in model 2 we performed a lag selection with interval and functions, that is, we apply a feature transformation; and model 3 looked at lag selection and nonlinear transformation. The individuals are represented as follows: in model 1 the individuals are:

$$P : \underbrace{[(T,F)_0, \ldots, (T,F)_{n-1}]}_{P_{fst}}$$

where only $T[i]$ takes values in $\{-1, maximumLag\}$; $T[i] = -1$ means that the attribute is not selected and $T[i] \neq -1$ means that the attribute i has a delay of $T[i]$. Spatial S and function F elements do not play any role in this case. In model 2 the individual is represented as:

$$P : \underbrace{[(T,F)_0, \ldots, (T,F)_{n-1}]}_{P_{fst}}$$

where $T[i]$ takes values in $\{-1, maxT\}$; $T[i] = -1$ means that the attribute is not selected and $T[i] \neq -1$ means that the attribute i has a delay of $T[i]$; and $F[i]$ takes values in {0,maxF}. Here the element

F codifies the way in which delays can act in the phenomena, for example a uniform, exponential or hyperbolic way. Finally, in model 3, the individual is represented as:

$$P : [\underbrace{(T,F)_0, \ldots, (T,F)_{n-1}}_{P_{fst}}]$$

where $T[i]$ takes values in $\{-1, maxT\}$; $T[i] = -1$ means that the attribute is not selected and $T[i] \neq -1$ means that the attribute i has a delay of $T[i]$; and $F[i]$ takes values in {0,maxF}. Here, F codifies the nonlinear transformation that is, the power to which each attribute is elevated. As a black box learning algorithm, we used the linear regression in all cases. In model 1, we ran in 5-fold *cross-validation* mode, with standard parameters and no embedded feature selection. While in the rest of models we ran in 10-fold *cross-validation* mode, also with standard parameters and no embedded feature selection. The optimization problem is a two-objective optimization problem in models 1 and 2 and a three-objective optimization problem in model 3. In models 1 and 2, the optimization equation looks like:

$$\begin{cases} \max Corr(\mathcal{V}(\mathcal{I}, P)), \\ \min Card_{fst}(P) \end{cases} \tag{5.1}$$

where, $Corr(\mathcal{V}(\mathcal{I}, P))$ refers to the correlation between the stochastic variable obtained by the observations and the linear variable obtained by linear regression on the data set $\mathcal{I}$ after the transformation indicated by P, and $Card_{fst}$ refers to the number of selected features. In model 3, we have two variants of a three-objective optimization equation, that is:

$$\begin{cases} \max Corr(\mathcal{V}(\mathcal{I}, P)), \\ \min Card_{fst}(P), \\ \min MaxExp(P) \end{cases} \tag{5.2}$$

where $MaxExp(P)$ measures the maximum exponent of the nonlinear transformation, and:

$$\begin{cases} \max Corr(\mathcal{V}(\mathcal{I}, P)), \\ \min Card_{fst}(P), \\ \min SumExp(P) \end{cases} \tag{5.3}$$

where $SumExp(P)$ measures the sum of all exponents of the nonlinear transformation.

5.4 Settings

In model 1, we used the standard mutation and crossover operations (suitably adapted to correctly deal with our solution representation), with probabilities (tuned with an initial experiment) of 0.3 and 0.7, respectively. Our population was composed of 100 individuals. We set the algorithm for a total of 1,000 evaluations in a single execution, and launched 5 independent executions, with seeds from 1 to 5. Moreover, we considered only one pollutant, NO_2. We considered only the 10% of the entire data set, and we split it into a training set and a test set, operating the optimization on the former only. The maximum lag (value of $T[i]$) allowed is 24 hours. In model 2, we used the first 30% of the data (taking into account both pollutants, NO_2 and NO_x). For each execution, NSGA-II was run with an initial, randomly generated population of 100 individuals, for 100 generations. Mutation and crossover have been suitably modified to deal with our individuals, and their relative probabilities have been left as originally set in the *jMetal* library. We launched 10 independent executions, with random seeds from 1 to 10; the maximum value of $T[i]$, $maxT$ was set to 24, and the maximum value of $F[i]$ was set to 3 including the uniform ($F[i] = 0$), linear ($F[i] = 1$) and hyperbolic ($F[i] = 2$) functions. Finally, in model 3, we initially partitioned our data into a training set (30%) and a test set (70%), respecting the temporal

		individual
exec.	*coeff.*	*a,d,w,h,p,t*
1	0.718	14,7,2,10,0,0
2	0.718	14,0,2,8,10,0
3	0.719	21,5,2,9,23,0
4	**0.721**	**20,0,2,7,19,0**
5	0.704	21,0,3,8,7,0

Table 5.3: Results, one line per execution, for NO_2 after performing model 1.

		individual					
exec.	*coeff.*	*a*	*d*	*w*	*h*	*p*	*t*
1	0.730	(10,1)	(1,0)	(2,0)	(7,0)	(2,1)	(1,0)
2	0.728	(0,1)	(7,1)	(2,0)	(9,0)	(4,1)	(2,0)
3	0.732	(22,1)	(2,0)	(5,0)	(7,0)	(0,1)	(2,0)
4	0.729	(2,0)	(12,0)	(2,0)	(11,0)	(1,1)	(2,0)
5	0.729	(1,1)	(14,0)	(2,0)	(8,0)	(0,0)	(2,0)
6	0.733	(6,0)	(2,1)	(3,0)	(10,1)	(21,0)	(2,1)
7	0.737	(23,1)	(2,1)	(4,1)	(8,1)	(3,0)	(3,1)
8	**0.732**	**(2,1)**	**(1,0)**	**(3,0)**	**(6,0)**	**(2,0)**	**(2,1)**
9	0.733	(2,1)	(2,1)	(3,0)	(9,0)	(11,0)	(2,0)
10	0.737	(22,1)	(2,1)	(3,0)	(8,1)	(1,1)	(2,0)

Table 5.4: Results, one line per execution, for NO_2 after performing model 2.

ordering, taking into account both NO_2 and NO_x pollutants. We performed 10 independent executions on the training data and then we chose the best-performing (and explainable) solutions and applied the corresponding transformation to the test data set. On each execution, we set the maximum value of $T[i]$ at 6 hours ($maxT = 6$), and the maximum value of $F[i]$ at 3 ($maxF = 3$).

5.5 Results

Table 5.3 shows the correlation coefficient of the best individual for each of the five executions for NO_2 after applying model 1. We also show the individual values for each variable, in the original order: air temperature, solar radiation, wind, relative humidity, pressure and traffic. Boldfaced values have the highest performance measure. Table 5.4 and 5.5 shows the results of the experiment with model 2. Results after performing model 3 are shown in Table 5.6 for NO_2, and in Table 5.7 for NO_x. Both tables are structured in the same way as the previous ones. In view of the results that have been obtained, two important elements must be considered: first, how lags and non-linear transformations are explained in the physical process and, second, how the transformations improve the synthesis of regression models. Regarding the first point, it is important to understand that different executions may give rise to different results and, yet, have similar performances. On one hand, some lags *are* consistent in all executions, which indicates that the temporal component of their contribution is stable and clear. On the other hand, some lags do show certain variability: we believe that in such cases, more experiments are necessary to fully understand the underlying mechanisms. All notwithstanding, it is clear how individuals have a clear positive effect on the cross-validated performances for linear regression. Observe both the average and the each single cross-validation execution the correlation coefficient of the models. In both cases, for NO_2 and NO_x, linear regression performed on the original data (that is, without any transformation) left a

		individual					
exec.	*coeff.*	a	d	w	h	p	t
1	0.629	(1,0)	(11,0)	(2,0)	(1,0)	(2,0)	(1,0)
2	0.628	(4,0)	(8,1)	(4,0)	(1,1)	(20,0)	(1,0)
3	0.629	(1,1)	(10,1)	(3,0)	(24,1)	(1,1)	(2,1)
4	0.628	(4,1)	(12,0)	(2,0)	(2,1)	(0,1)	(1,0)
5	**0.638**	**(2,1)**	**(10,0)**	**(4,0)**	**(2,0)**	**(23,1)**	**(1,0)**
6	0.632	(0,1)	(12,1)	(3,0)	(1,1)	(7,1)	(1,0)
7	0.632	(1,0)	(9,0)	(4,0)	(1,1)	(2,0)	(2,1)
8	0.630	(2,1)	(13,0)	(2,1)	(1,0)	(1,1)	(2,1)
9	0.631	(1,0)	(10,0)	(2,0)	(2,0)	(4,0)	(2,1)
10	0.630	(1,0)	(9,0)	(2,0)	(2,1)	(14,1)	(1,0)

Table 5.5: Results, one line per execution, for NO_x after performing model 2.

		individual					
exec.	*coeff.*	a	d	w	r	p	t
1	0.727	(0,3)	(1,1)	(2,1)	(5,1)	(5,1)	(1,1)
2	0.725	(5,2)	(0,1)	(2,1)	(5,1)	(1,2)	(1,1)
3	0.725	(6,1)	(1,1)	(2,1)	(6,1)	(4,1)	(1,1)
4	0.723	(5,3)	(6,3)	(2,1)	(5,3)	(0,3)	(1,1)
5	0.728	(0,3)	(0,1)	(2,1)	(6,1)	(0,1)	(1,1)
6	0.729	(0,2)	(1,1)	(2,1)	(6,1)	(0,1)	(1,1)
7	0.732	(1,3)	(1,1)	(2,1)	(6,1)	(1,1)	(1,1)
8	0.725	(4,3)	(4,2)	(2,1)	(6,1)	(5,1)	(1,1)
9	**0.733**	**(6,3)**	**(1,2)**	**(2,1)**	**(6,1)**	**(0,1)**	**(1,1)**
10	0.729	(3,3)	(0,1)	(2,1)	(6,2)	(3,1)	(1,1)

Table 5.6: Results, one line per execution, for NO_2 after applying model 3.

correlation coefficient of 0.63 and 0.61 respectively. It is clear that linear regression on the transformed data improved the coefficients. Moreover, we can observe as higher performances are obtained when lag combinations or nonlinear transformations are applied. By looking at each variable separately, consider the following. Traffic influences the amount of pollutant concentrations in a positive way, and with one hour of delay. This delay can be explained by the fact that the effect of exhaust gases need some time to accumulate. Also observe that we are limited by the granularity of observations: if the sensors collected data with granularity, for example, one minute, we may have seen shorter delays for this variable. Higher air temperatures are associated with a decrease in NO_2 and NO_x concentrations, which is caused by three processes: at higher air intake temperatures, less NO_2 is produced in the process of fuel combustion in the engine; *(i)* higher air temperatures usually imply more favorable atmospheric conditions, which encourages residents to use alternative means of transportation which reduces traffic volume [146], and *(ii)* NO_2 is dynamically transformed into NO (and back) at a higher rate with higher temperatures. Wind always affects concentration in a negative way, with a constant (across different executions) delay of two hours. This is likely due to the distance between the intersection and the meteorological station; the average wind speed of 3 ms^{-1}, as well as the roughness of terrain explains the amount of the delay. Higher humidity may cause a decrease in NO_2 concentration in exhaust gases (as observed in [147, 148]). We have found that such an effect takes place with 5-6 hours of delay. High solar duration and high

		individual					
exec.	*coeff.*	*a*	*d*	*w*	*r*	*p*	*t*
1	0.614	(2,1)	(5,2)	(2,1)	(0,1)	(6,1)	(1,1)
2	**0.619**	**(1,1)**	**(6,1)**	**(2,1)**	**(1,1)**	**(4,1)**	**(1,1)**
3	0.618	(1,1)	(6,1)	(2,1)	(1,2)	(6,3)	(1,1)
4	0.613	(2,2)	(6,2)	(2,1)	(1,1)	(5,1)	(1,1)
5	0.615	(0,1)	(5,1)	(2,1)	(1,1)	(3,1)	(1,1)
6	0.616	(2,1)	(6,1)	(2,1)	(0,1)	(3,1)	(1,1)
7	0.614	(1,1)	(6,1)	(2,1)	(0,1)	(0,1)	(0,1)
8	0.615	(1,1)	(5,1)	(2,1)	(1,2)	(5,1)	(1,1)
9	0.618	(0,1)	(6,1)	(2,1)	(1,1)	(6,1)	(1,1)
10	0.616	(2,1)	(5,1)	(2,1)	(1,1)	(0,1)	(1,1)

Table 5.7: Results, one line per execution, for NO_x after applying model 3.

temperature favor the transformation of nitrogen oxides into secondary pollutants, which includes ozone, and this implies a decrease in NO_x concentration. Even more important is sunlight, which acts as a catalyst and explains the negative coefficient and the delay for sunlight duration.

Chapter 6

Application 3: Antique Buildings

Materials research applied to the study of archaeological ceramic artifacts is now a consolidated practice, and during the last decades the interest of scientists, architects, engineers and archaeologists towards architectural heritage preservation has risen. The ancient buildings within historic city centres mark the image and history of each city at different periods. When damaged these buildings become damaged, the restoration of historic masonry is needed. A good characterization of both new and old material is crucial to predict the behavior of the system. Historical understanding is not only useful to analyze and preserve objects but also to investigate the knowledge and skills used to produce and use them. In this light, the main goals of the building materials' characterization are preservation and restoration, which include [149]: *(i)* origin of historical raw materials, *(ii)* processes and changes in archaeological artifacts, *(iii)* determination of original firing temperature, and *(iv)* reconstruction of firing techniques and manufacturing technologies. Bricks and ceramics can be considered artificial rocks fired in kilns. Under this point of view, mineralogical, petrological and geochemical study approaches can be useful tools for the study of archaeological ceramic materials. Bricks in ancient buildings preserve the trace of the claystone geological formation used to create them and geology influences raw material availability and, thus, building methods. This chapter considers the problem of establishing a chemical fingerprint of antique buildings in the city of Ferrara, Italy. We define this problem within our framework as a *feature selection and transformation* for classification problem. This chapter is based on [150].

6.1 Data

The data used for this study consists of 113 samples of clays and bricks from nine different antique palaces located in the province of Ferrara. These included the Monastery of Sant'Antonio in Polesine (built in several phases during the 12th -16th centuries), the Church of Santa Maria in Vado (founded in the 10th century and extensively modified in the 15th-16th centuries), the Church of Santo Stefano (founded in the 10th century, but rebuilt in the 15th -16th centuries), the Cathedral of Ferrara (apse, 15th-16th centuries), the Schifanoia Palace ('Hall of Stuccoes', 15th-16th centuries), the surrounding city walls (15th century), the Palazzo Roverella (built in the 16th century), the Monastery of Santa Maria delle Grazie (built in several phases starting from the 14th century) and the church of San Andrea (built around the year 1000, in the east part of the Byzantine Castrum, but what currently remain of the ancient splendor is only the aisle with small semicircular apses and a small portion of the apse remain). This nine-building compound is the the class attribute of our data matrix. The chemical variables, which are the independent attributes of our data matrix, are: TiO_2, Al_2O_3, Fe_2O_3, MnO, MgO, CaO, Na_2O, K_2O, P_2O_5, V, Cr, Ni, Cu, Zn, Ga, Rb, Sr, Zr, Ba, La, Ce, Pb, Sc, Co, Th and Y. The source in this case is equal to the class, therefore, our data matrix has no named attribute. Moreover, we have no

feature	*mean*	*p-value*	*kurtosis*	*skewness*
SiO_2	52.27	0.023	$4.08 * 10^{+14}$	-0.32
TiO_2	0.68	0.576	$2.68 * 10^{+14}$	-0.11
Al_2O_3	14.34	0.749	$2.62 * 10^{+13}$	-0.09
Fe_2O_3	5.93	0.636	$2.71 * 10^{+14}$	0.17
MnO	0.12	0.105	$2.71 * 10^{+14}$	0.04
MgO	4.45	$0.921 * 10^{-3}$	$3.81 * 10^{+14}$	-0.73
CaO	10.80	0.555	$3.70 * 10^{+13}$	-0.02
Na_2O	1.47	1.349	16.46	$3.14 * 10^{+14}$
K_2O	2.51	9.433	24.34	$3.51 * 10^{+14}$
P_2O_5	0.24	0.003	3.33	0.71
LOI	6.96	0.013	2.25	0.29
Pb	43.47	0.729	17.60	$3.36 * 10^{+12}$
Zn	114.45	$1.273 * 10^{-7}$	75.39	$8.02 * 10^{+14}$
Ni	146.47	$0.564 * 10^{-3}$	4.21	-0.70
Co	19.49	0.212	3.00	-0.25
Cr	205.36	0.0428	3.34	-0.55
V	102.53	0.001	5.89	0.88
Th	9.84	0.322	3.15	-0.23
Nb	8.49	1.237	-1.417	0.289
Zr	134.68	0.023	2.79	-0.46
Rb	118.15	0.576	52.23	$5.62 * 10^{+14}$
Sr	297.80	0.749	3.87	-0.18
Ba	417.82	0.636	25.51	$3.10 * 10^{+14}$
Y	25.34	0.105	4.278	-1.792
La	31.22	$0.921 * 10^{-3}$	3.18	0.63
Ce	52.26	0.555	4.33	0.58

Table 6.1: Descriptive statistical analysis of the data.

temporal or spatial information, as such, the data matrix has no temporal or spatial attributes.

6.2 Statistical Analysis

Some of the relevant statistical measures of our variables are shown in Table 6.1. They can be classified by their behavior in three major groups according to a Shapiro normality test: *normal* variables (that follow a normal distribution), that is, TiO_2, Al_2O_3, Fe_2O_3, MnO, CaO, Na_2O, K_2O, Pb, Co, Th, Rb, Sr, Ba and Ce; *quasi normal* variables (that do not follow a normal distribution, but with low values of skewness and kurtosis), that is: P_2O_5, LOI, Cr, La, V and Zr; *non-normal* variables: SiO_2, MgO, Zn, Ni and La. We show in Figure 6.1, Figure 6.2, Figure 6.3 and Figure 6.4 the graphical representation of the statistical behavior of each of the variables and the correlation between the elements can be seen in Table 6.2.

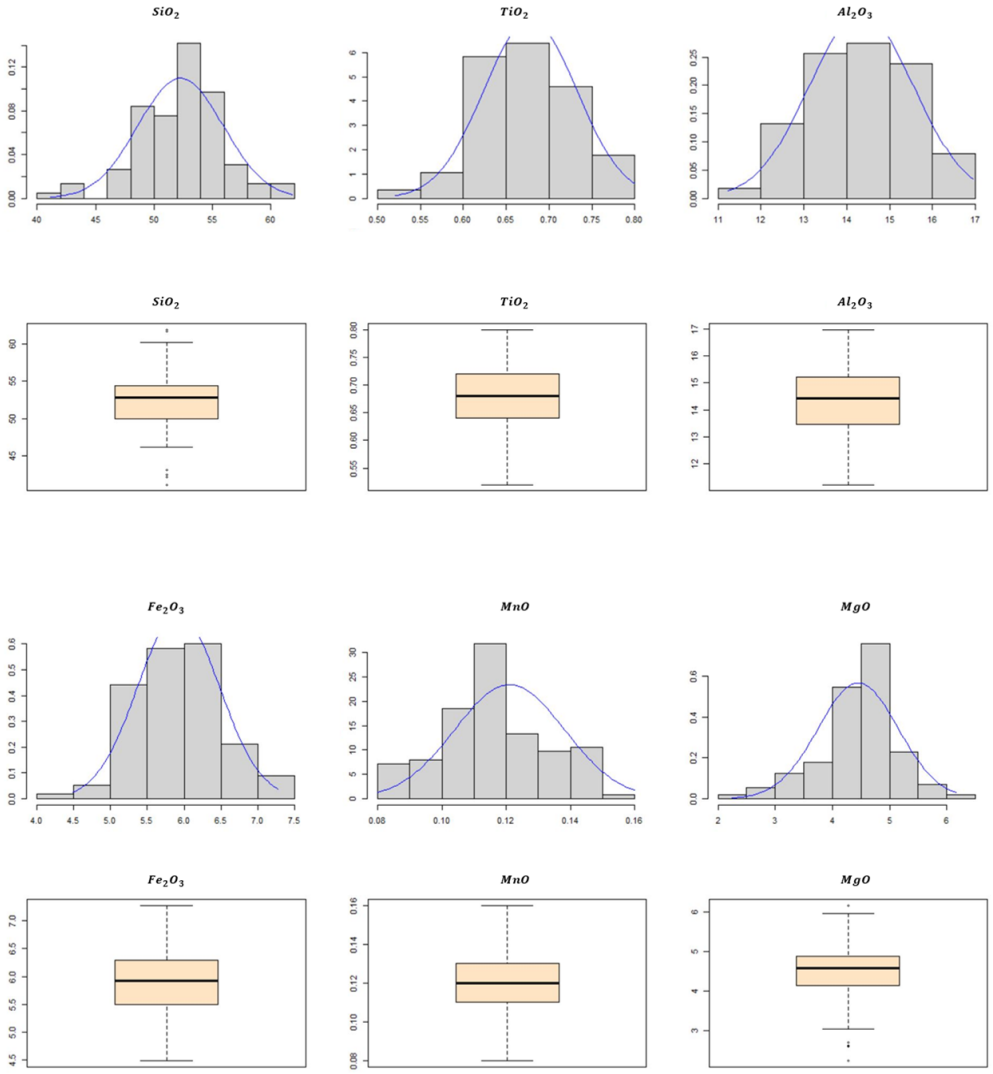


Figure 6.1: Distribution and outliers detection analysis: SiO_2,TiO_2,Al_2O_3,Fe_2O_3, MnO,MgO.

	Ni	Co	Cr	V	Th	Nb	Zr	Rb	Sr	Ba	Y	La	Ce	
	1	0.699	0.801	0.432	0.441	-0.210	-0.324	0.097	-0.124	0.137	0.070	0.061	0.339	Ni
SiO_2	1	1	0.582	0.341	0.371	-0.020	-0.304	0.138	-0.047	0.106	0.015	0.057	0.379	Co
TiO_2	0.453	1	1	0.544	0.354	0.040	-0.312	0.047	-0.255	0.178	-0.087	-0.011	0.347	Cr
Al_2O_3	0.408	0.860	1	1	0.160	0.26	-0.113	0.048	-0.223	0.193	-0.037	0.012	0.295	V
Fe_2O_3	0.267	0.932	0.846	1	1	-0.180	0.001	0.073	0.190	0.129	0.255	-0.030	0.303	Th
MnO	0.086	0.342	0.176	0.373	1	1	0.106	-0.189	-0.416	-0.069	-0.094	0.186	-0.02 5	Nb
MgO	0.046	0.436	0.504	0.415	0.012	1	1	0.038	0.272	-0.214	0.658	-0.018	-0.102	Zr
CaO	-0.516	-0.088	-0.263	-0.029	0.168	-0.157	1	1	0.222	0.134	-0.040	-0.088	0.154	Rb
Na_2O	-0.232	-0.328	-0.408	-0.312	-0.036	-0.228	0.017	1	1	-0.007	0.461	-0.142	0.015	Sr
K_2O	-0.060	-0.187	-0.132	-0.048	-0.040	-0.126	-0.232	0.217	1	1	-0.090	-0.128	-0.038	Ba
P_2O_5	0.073	0.188	0.270	0.206	-0.110	0.066	0.228	-0.396	-0.206	1	1	-0.047	0.098	Y
LOI	-0.677	-0.678	-0.068	-0.591	-0.271	-0.279	-0.007	0.267	0.067	-0.172	1	1	-0.037	La
Pb	0.198	0.047	0.728	0.027	-0.019	-0.141	-0.121	-0.089	-0.140	0.219	-0.055	1	1	Ce
Zn	0.314	-0.169	0.693	-0.115	-0.053	0.029	-0.406	-0.044	0.621	-0.098	-0.034	0.102	1	
	SiO_2	TiO_2	Al_2O_3	Fe_2O_3	MnO	MgO	CaO	Na_2O	K_2O	P_2O_5	LOI	Pb	Zn	

Table 6.2: Correlation matrix.

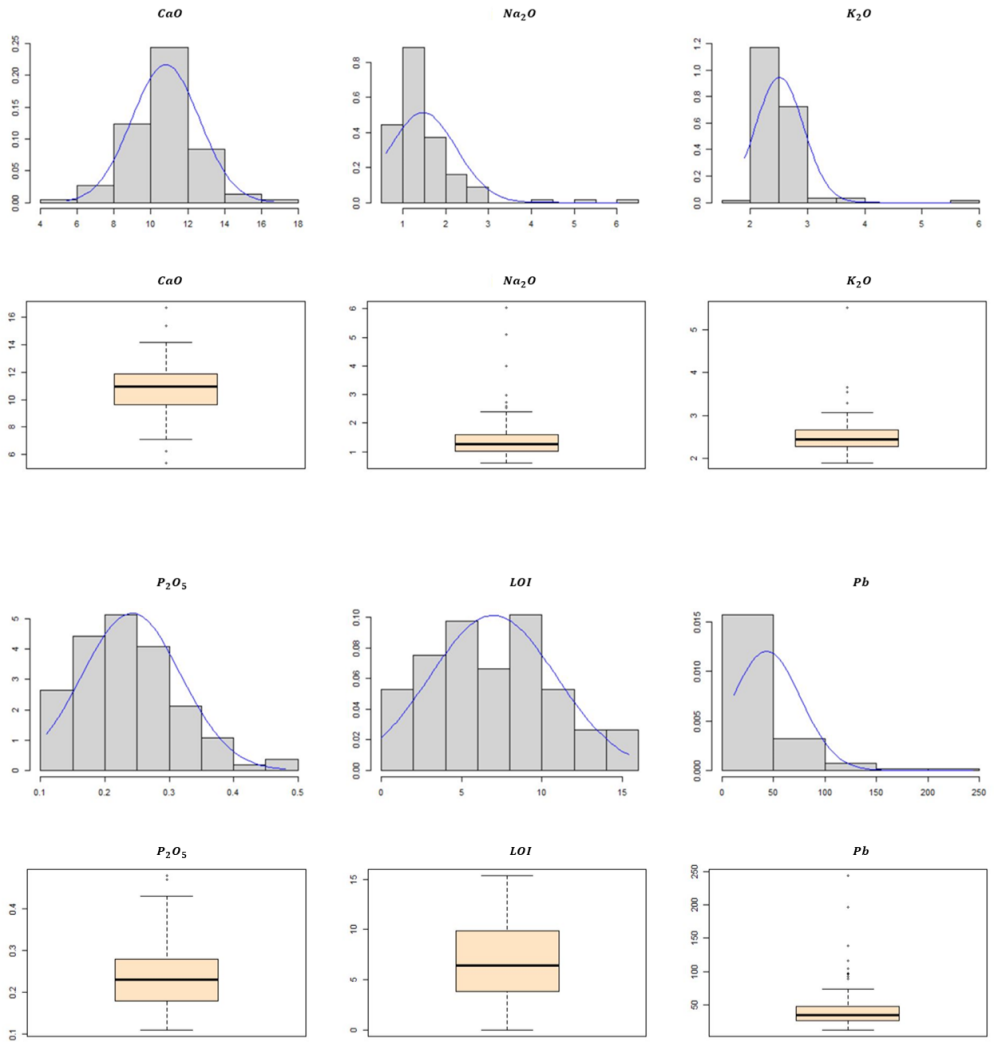


Figure 6.2: Distribution and outliers detection analysis: CaO,Na_2O,K_2O,P_2O_5, LOI,Pb.

6.3 Learning Schemata

The typical *by hand* process for fingerprinting is very expensive and entails an elevated risk of inaccuracy due to potential loss of information, manual loading of data, and prolonged analysis time. In recent times, several statistical methods have been applied to aid the traditional geochemical investigation in understanding, for example, pollution sources via fingerprinting, possible correlation among elements and, in some cases, the nature of the contamination [151–153]. Examples of fingerprinting include recent works focused on the protection of groundwater against pollution and deterioration. Input pollution identification includes applying geographical information systems and decision analysis [154,155], logistic regression model learning [156], univariate and multivariate analysis [157], and multiple regression models [158]. In this work, we define a model to solve the geochemical fingerprint problem based on our general framework that consists of a feature selection with nonlinear transformation. The individual is represented as follows:

$$P : [\underbrace{(T,F)_0,\ldots,(T,F)_{n-1}}_{P_{fst}}]$$

where $T[i]$ takes values in $\{-1,0\}$; $T[i]=-1$ means that the attribute is not selected and $T[i]=0$ means

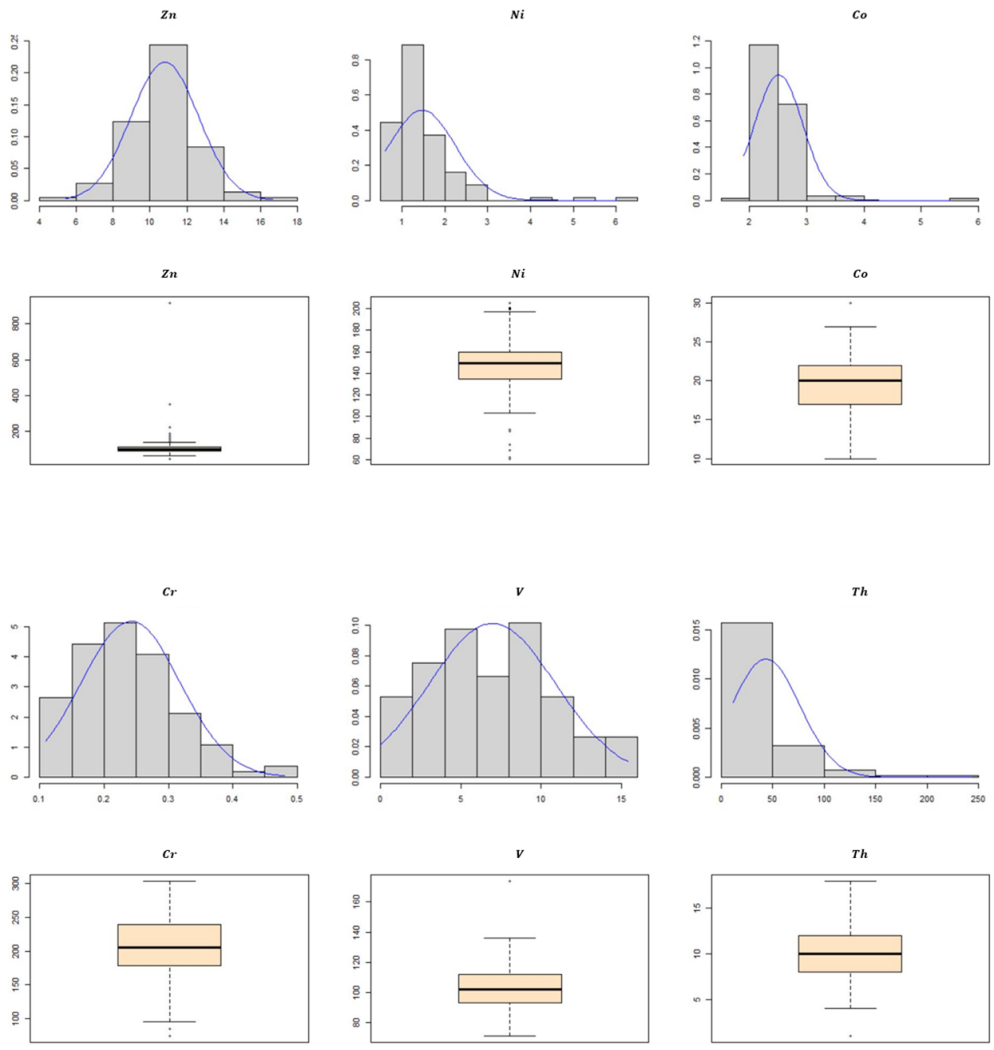


Figure 6.3: Distribution and outliers detection analysis: Zn,Ni,Co,Cr, V,Th.

that the attribute i is selected. $F[i]$ takes values in $\{0, maxF\}$ where $F[i] = 0$ means that the attribute i is not elevated to any power. $F[i] \neq 0$ means that the attribute i is elevated to the power $F[i]$. Utilizing a black box learning algorithm, we used three different classifiers in order to determine which type of classification shows the best performances for this particular problem. In particular we used a *single decision tree* (J48), a *random forest* (RF), and a *logistic regression algorithm* (LR). Thus, as performance measures, the accuracy and the function are defined as:

$$Acc(\mathcal{V}(\mathcal{I}, P))$$

Finally, the following optimization equation used was:

$$\begin{cases} \max Acc(\mathcal{V}(\mathcal{I}, P)), \\ \min Card_{fst}(P), \\ \min MaxExp(P) \end{cases} \tag{6.1}$$

where, as in previous applications, $MaxExp(P)$ measures the maximum exponent of the nonlinear transformation and $Card_{fst}$ the number of selected features.

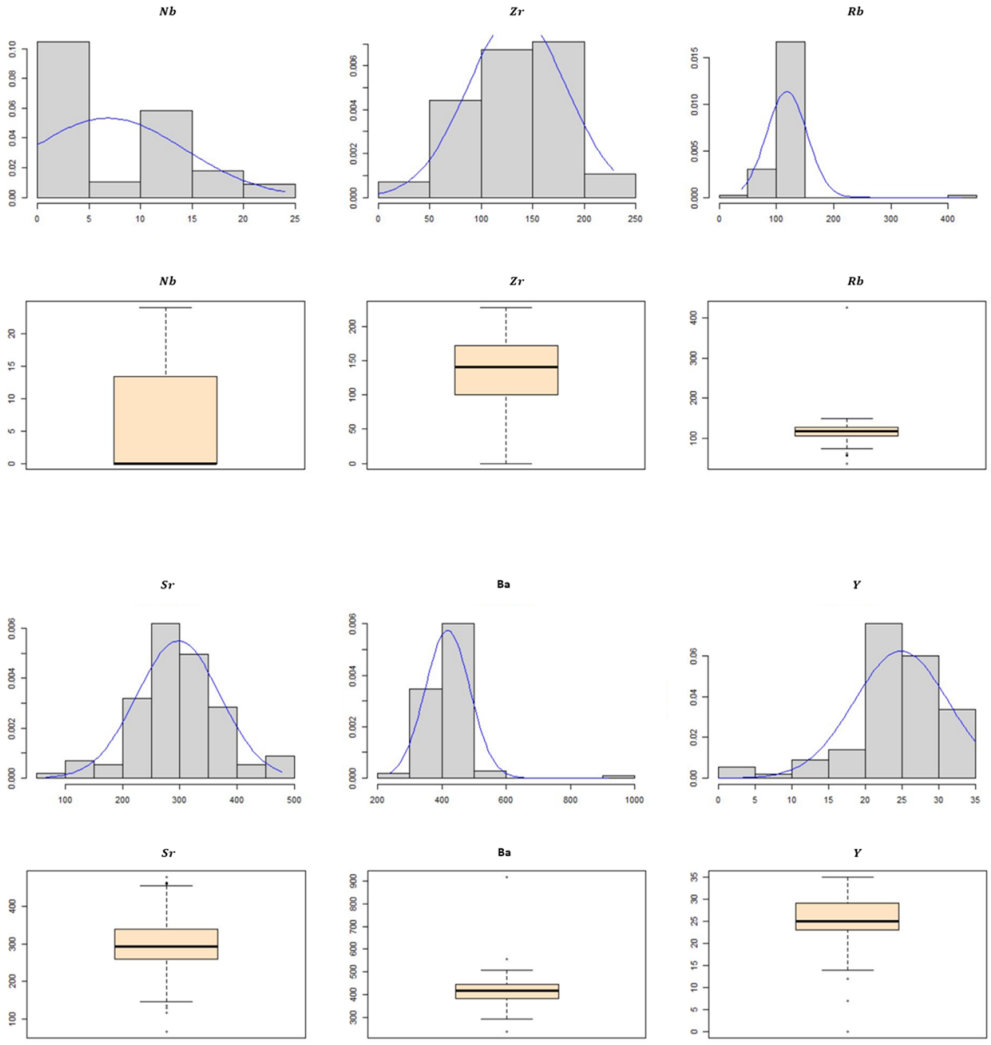


Figure 6.4: Distribution and outliers detection analysis: Nb,Zr,Rb,Sr, Ba,Y.

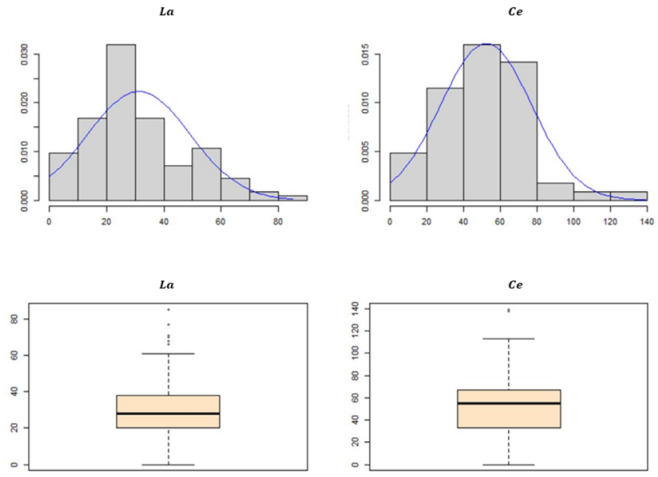


Figure 6.5: Distribution and outliers detection analysis: La,Ce.

6.4 Settings

We ran two experiments, one using all chemical variables and another adding the following ratios of variables: MgO/Cr, SiO_2/TiO_2, SiO_2/Al_2O_3, SiO_2/CaO, SiO_2/Na_2O, SiO_2/K_2O and Cr/Ni. All classifiers were used with their standard parametrization. To evaluate them, we used the so-called *leave-one-out cross validation* ($LOOCV$), which is a variant of the more general *cross validation* (CV). In cross validation, we randomly divided a dataset into k disjoint folds of approximately equal size, and each fold is then used to test the model induced from the other $k-1$ folds by a classification algorithm. The performance of the classification algorithm is evaluated by averaging the performances of the k models, and hence the level of averaging is assumed to be at fold. The leave-one-out version is used to estimate the performance of a classification algorithm by setting k to be the number of examples in the dataset, making it the best option to be used in the case of small datasets, and is able to produce robust enough estimates of model performance as each instance is given an opportunity to represent the entirety of the test dataset. The maximum function allowed was set to three and we limited the fingerprint lenght to less than six variables. We used the evolutionary algorithm, NSGA-II with the standard parameters, in each experiment with standard mutation and crossover. We used an initial population of 10 individuals, each evolving 1,000 times. Each experiment was run 10 times and the best absolute solution was considered, in terms of accuracy of the classification model.

6.5 Results

Table 6.3 and Table 6.4 display the results of the experiment without ratios variables. In particular, we show the most accurate fingerprints per each execution. Table 6.4 shows the shorter fingerprint and we can initially conclude that the fingerprinting problem *per se* can be solved, as the average accuracies in leave one out cross-validation mode range from 65% to 81%. More specifically, we also observe that interpretable and quasi-interpretable models seem to work slightly better than non-interpretable models. In this case and considering this, across all selected fingerprints, their average accuracy is slightly higher. Table 6.5 and Table 6.6 show the results of the experiment including ratios variables. In some cases, We observe that the accuracies are higher while also maintaining an interpretable fingerprint. Finally, our approach can be completed with a graphical account of selected solutions. We show two examples in Figure 6.6 and 6.7, in which we displayed only the buildings that separate best from each other. As observed, separating lines are highly non-linear: this can be taken as empirical proof that non-automatized approaches to this problem are not feasible, as fingerprints cannot be easily noticed by a visual examination of the data.

classifier	*fingerprint*	*acc.*
J48	$SiO_2{}^3$, $Al_2O_3{}^3$, Na_2O^3, $P_2O_5{}^2$, Th^2, Nb^3, Zr, Rb^3, Y^2	0.88
	$P_2O_5{}^2$, LOI, Zn^2, Cr, V^3, Nb^3, La^3	0.86
	Fe_2O_3, Na_2O^3, Ni, Co, Th^3, Nb^2, Zr, Rb, Y	0.86
	$SiO_2{}^3$, MgO^3, Na_2O^3, K_2O^2, Nb, Zr^3, Rb, Sr, Y^3	0.86
	SiO_2, MnO^3, $P_2O_5{}^3$ V, Nb, Ce	0.86
RF	MgO^2 ,Na_2O^3 ,LOI^2, Pb^2, V, Nb^2 , Zr^3, Sr^3, Y^3	0.87
	K_2O^2, $P_2O_5{}^2$, LOI^2, Nb^2, Zr, Sr^2, Ba, Y^3, La^3	0.88
	$Al_2O_3{}^2$, $P_2O_5{}^3$, Zn, V^3, Nb^2, Rb^3, Sr^2, La^3	0.86
	Na_2O, $P_2O_5{}^3$, Pb, Zn, Co^2, Nb^3, Zr^3, Ba^2	0.88
	MnO, MgO^2, Na_2O^3, $P_2O_5{}^3$, LOI^3, Th^2, Nb^2, Zr^2, Rb^2	0.90
LR	Fe_2O_3, Na_2O, K_2O, LOI^3, Zn, Ni^3, Nb^3, Sr^2, Ba^3, La^3, Ce	0.84
	$SiO_2{}^2$, Al_2O_3, MnO^2, MgO, $P_2O_5{}^3$, LOI, Ni, Th^3, Nb^2, Sr^3, Y, Ce^3	0.86
	TiO_2, Al_2O_3, P_2O_5, Zn, Co^3, Nb^2, Zr^3, Sr^3, Y^2, Ce^2	0.81
	$TiO_2{}^2$, MnO, Na_2O^2, P_2O_5, LOI^2, Ni, V^2, Rb, La^2	0.82
	$SiO_2{}^3$, $TiO_2{}^3$, MgO^2, Na_2O, LOI, Zn, Co^3, Nb^3, Zr, Rb^2, La^3	0.86

Table 6.3: Results of the major and trace elements fingerprinting experiments.

classifier	*name*	*fingerprint*	*acc.*
J48	f1	Al_2O_3, $P_2O_5{}^2$, Nb^2, Y^2	0.83
	f2	MgO, Na_2O^3, $P_2O_5{}^2$, Nb, Zr^2	0.87
	f3	$TiO_2{}^2$, $P_2O_5{}^3$, Nb, Sr^3	0.82
	f4	Na_2O, P_2O_5, Nb^2, Rb^2, Y^2	0.86
	f5	$Al_2O_3{}^2$, MgO, $P_2O_5{}^2$, Nb^3, Ce^3	0.83
RF	f6	$Fe_2O_3{}^3$, Na_2O, $P_2O_5{}^3$, Nb^3, Zr^3, Ba	0.87
	f7	MgO^3, CaO^3, Na_2O^3, Nb, Ce^3	0.86
	f8	$Fe_2O_3{}^3$, MnO, Zn^3, Nb^2, Y^3, La	0.86
	f9	Na_2O^2, $P_2O_5{}^2$, Nb, Zr^3, Rb^3	0.87
	f10	SiO_2, CaO^3, Na_2O^2, Nb^3, Zr^2, Rb, Ce^2	0.88
LR	f11	Na_2O, Pb, Co^3, V, Nb, Zr^2	0.82
	f12	K_2O, Pb^2, Zn, Nb^3, Sr^3, Y^2, Ce^3	0.80
	f13	Na_2O^2, $P_2O_5{}^2$, Nb, Zr^3, Rb^3	0.85
	f14	$Al_2O_3{}^2$, Na_2O^2, K_2O^3, $P_2O_5{}^2$, Nb^2, Zr^3, Ba, La^3	0.83
	f15	$TiO_2{}^3$, Fe_2O_3, MgO^2, CaO^3, Na_2O, Nb^3, Rb, Ce^2	0.83

Table 6.4: Results of the shortest fingerprinting.

classifier	*fingerprint*	*acc.*
J48	$SiO_2{}^3$, TiO_2, K_2O^2, Nb^3, Zr^2, Ce^2, MgO/Cr^2, SiO_2/Na_2O^2	0.87
	SiO_2, K_2O, Co^2, V^2, Nb^3, Rb^3, $SiO_2/Al_2O_3{}^3$, SiO_2/Na_2O^2	0.88
	Na_2O^2, Nb^3, Zr, Rb, Sr^2, Y^3, SiO_2/K_2O	0.87
	$TiO_2{}^3$, K_2O^3, Nb^2, Rb, Y, SiO_2/Na_2O, SiO_2/K_2O^3	0.85
	$Fe_2O_3{}^2$, Na_2O^2, Nb, Rb^3, Ba^2, SiO_2/TiO_2, SiO_2/CaO^3	0.87
RF	SiO_2, $Fe_2O_3{}^2$, LOI, Nb^2, Zr^2, Ce, MgO/Cr^2, SiO_2/TiO_2	0.85
	$P_2O_5{}^2$, Co, Cr, V^3, Nb, Zr, Rb^2, La^2, SiO_2/K_2O^2	0.88
	Al_2O_3, $Fe_2O_3{}^2$, Nb^3, Rb^3, La^2, MgO/Cr^3, $SiO_2/TiO_2{}^2$, SiO_2/Na_2O^3	0.89
	Fe_2O_3, Na_2O, LOI, Pb^2, V, Nb^2, Zr^3, Sr^2, $SiO_2/Al_2O_3{}^3$	0.88
	SiO_2, Na_2O^2, Ni, Co^3, Nb, Ba^2, MgO/Cr^3, SiO_2/TiO_2	0.86
LR	TiO_2, $Al_2O_3{}^2$, MgO^3, Na_2O^2, Pb, Nb^2, Y, La, MgO/Cr, SiO_2/TiO_2, SiO_2/CaO, SiO_2/K_2O, Cr/Ni	0.84
	CaO^2, LOI^3, Zn^2, V^2, Nb^3, Zr^2, Rb, Ba^3, Ce, SiO_2/Na_2O^3, SiO_2/K_2O	0.88
	$Al_2O_3{}^3$, CaO^2, $P_2O_5{}^3$, LOI^2, Pb^2, Zn^3, Nb, Zr^3, Rb^2, Ba^2, SiO_2/TiO_2, SiO_2/Na_2O, SiO_2/K_2O	0.91
	$SiO_2{}^2$, MnO^2, Na_2O, $P_2O_5{}^3$, Cr^2, Nb, Y^2, Ce^3, MgO/Cr^2, SiO_2/CaO^2	0.87
	$SiO_2{}^2$, Al_2O_3, MnO^2, MgO^2, CaO, Na_2O, LOI^3, Nb, Zr, Sr, Ce^2, SiO_2/K_2O^3	0.89

Table 6.5: Results of the major and trace elements ratios-based fingerprintings.

classifier	*fingerprint*	*acc.*
J48	P_2O_5, Ni^2, Nb, SiO_2/Al_2O_3	0.84
	$SiO_2{}^3$, Na_2O^2, P_2O_5, Nb^3, SiO_2/TiO_2	0.85
	Pb, Nb^3, Y^2, SiO_2/Na_2O	0.97
	MnO^2, P_2O_5, Ni, Nb^2, SiO_2/TiO_2	0.85
	Na_2O^2, Nb^3, Zr^2, Y^3, Cr/Ni^2	0.85
RF	SiO_2, $Al_2O_3{}^2$, V, Nb, Rb, SiO_2/Al_2O_3	0.80
	Pb^2, Co^3, Cr^2, Nb^3, Ce, SiO_2/TiO_2, SiO_2/Na_2O^2	0.87
	MnO^2, Zn, Ni^3, Nb^2, Ce^2, SiO_2/CaO^2, SiO_2/K_2O^3	0.96
	Na_2O^2, P_2O_5, La^3, $SiO_2/Al_2O_3{}^2$, SiO_2/CaO^3, SiO_2/K_2O^3	0.83
	K_2O^3, V^3, Nb, Ce^3, $SiO_2/TiO_2{}^2$, $SiO_2/Al_2O_3{}^2$, SiO_2/Na_2O	0.87
LR	MgO, $P_2O_5{}^2$, Nb, Zr, Y, Ce^2, SiO_2/K_2O^2	0.83
	P_2O_5, LOI^2, Zn, Nb^3, La^3, $SiO_2/Al_2O_3{}^3$, SiO_2/Na_2O^3	0.86
	$P_2O_5{}^2$, Ni, Nb^2, SiO_2/Al_2O_3	0.82
	$P_2O_5{}^3$, V, Nb, Zr^3, Sr^2, $SiO_2/TiO_2{}^2$, Cr/Ni	0.86
	$TiO_2{}^2$, Nb, Sr, Y^2, Ce, SiO_2/Al_2O_3	0.84

Table 6.6: Results of the shortest ratios-based fingerprintings.

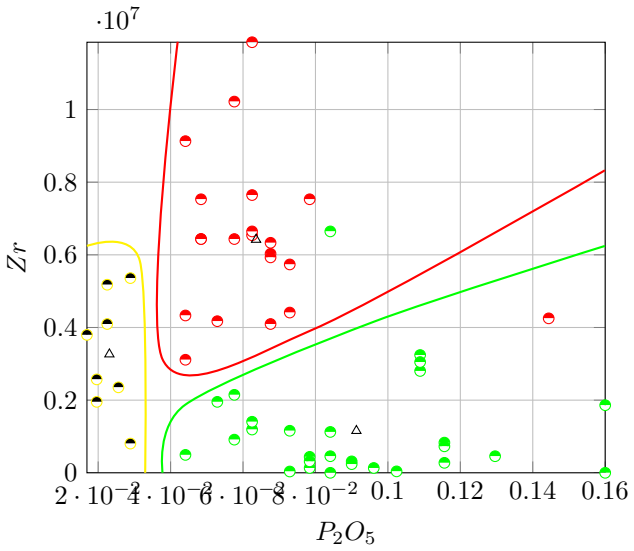


Figure 6.6: Fingerprint representation of solution 12 (Duomo building in red, Pietro building in green and Roverella building in yellow; centroids are represented with a triangle)

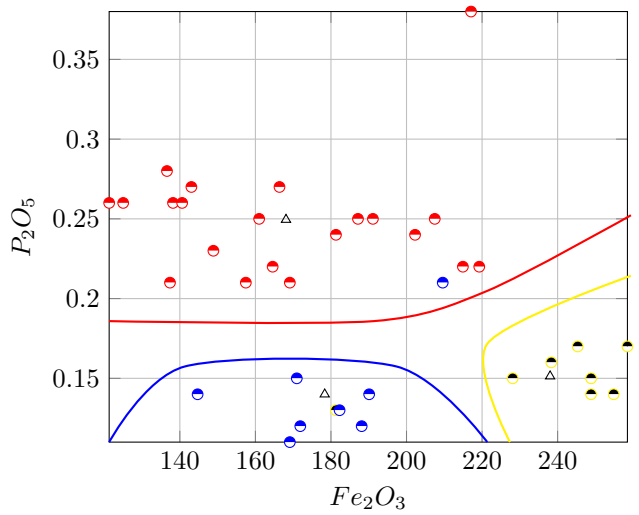


Figure 6.7: Fingerprint representation of solution 1 (Grazie building in blue, Duomo building in red and Roverella building in yellow; centroids are represented with a triangle).

Chapter 7

Conclusions

The starting point of this work was the observation that many statistical learning problems share common characteristics in both data presentation and learning objectives. Because of this, as a first step, we introduced a generalized data set syntax which highlighted the roles of attributes, the temporal and the spatial components, the class, and the names. Inspired by previous research, we then identified a series of *meta-problems* of interest to the real world by learning from data, including fingerprinting, temporal lag identification, non-linear transformations, and spatial displacements. We separated these meta-problems from the learning problems and we proposed a generalized methodology for their solution. Such a methodology is mathematically well-founded and formalized, reproducible, and specifically designed to treat data of natural origin. Our contributions can be summarized as follows.

1. We first surveyed and systematized several recent scientific contributions to the field of data preprocessing, in an attempt to highlight their common nature (Chapter 2). Thus, we give a general vision of preprocessing data techniques that meta-problems require, the current availability of preprocessing machine learning methods and what each preprocessing technique aims to solve.

2. Then we designed a mathematical framework that generalized most preprocessing techniques, separated them from the learning phase, and allowed for each meta-problem to be individually addressed (Chapter 3). As a result of such a separation, the result benefits the interpretability, even when non-interpretable learning algorithms are used. Here, the principal element is the individual of the multi-objective evolutionary algorithm and it serve as pivot of the mathematical framework. The characteristics of the package of the multi-objective evolutionary algorithm allow an easy implementation of all components according to the individual definition. And the individual definition is the key of the interpretability of the results.

3. Finally, we tested our techniques in three different, real cases. Data came from very different sources and application domains (groundwater wells, air pollution monitoring stations, and antique buildings clay samples). We fitted the problems that were proposed on such data as instances of our dynamic preprocessing technique, and, after analysing the solutions, we found that they were always better, and more interpretable, than those that could be devised with standard methods (Chapter 4, 5, and 6).

There are many questions that remain unanswered that could be addressed as future research. Firstly, our definition of data set gives rise to a whole learning discipline characterized by instances of dimensional components and individual classes. Thus, we assume that we want to classify (or regress) each instance individually, but we want to take into account its 'surroundings' (either temporal, spatial, or both). The next step in this generalization, is classifying (regressing) names, instead of single instances. Here, the entire set of instances is generally considered to learn a hidden phenomenon or process. However, in

some cases in which instances belong to different sources, for example the clinical case in which instances are associated with different patients, a learning task must by applied to the set of instances associated with each (patient) name. In addition, the dynamic preprocessing methodology should be systematically compared with its analogous methodology in terms of filters, and their differences and similarities should be noted which, would enable a better understanding of them. Moreover, the entire problem could be approached from the optimization algorithm point of view, thus answering the question of which optimization algorithm, and implementation, is best to this end. But more important, and certainly more challenging, is the question of how meta-problems and their solutions can be made available to the end user. Inspired by widespread, open-access tools such as Weka, which offer some form of graphical user interface and allows non-experts to tackle simple data science problems on their own, one can imagine a similar, but more specific, *fingerprinting, transformation, and cleaning* tool that would allow bio-geo-chemical experts to pose and solve their own meta-problems, and interpret the solutions. In such a tool, one could directly ask the question of *what is the geochemical fingerprint of this water sample*, or *food samples*, or *cultivation*, for example. Therefore, considering the possibility of cutting out the middle man and focusing on real-world problems, we believe, should be the purpose of artificial intelligence: to avoid repetitive questions or tasks and let the machine do all systematic work, which becomes non-intelligent (because a machine is able to perform it), therefore freeing experts' time and resources to focus on the next intelligent question.

Bibliography

[1] E. Lucena-Sanchez, G. Sciavicco, and I. E. Stan. Symbolic learning with interval temporal logic: the case of regression. In *OVERLAY*, pages 5–9, 2020.

[2] E. Lucena-Sánchez, G. Sciavicco, and I. E. Stan. Feature and language selection in temporal symbolic regression for interpretable air quality modelling. *Algorithms*, 14(3):76, 2021.

[3] F. Chávez-Castrillón, M. Coltorti, R. Ivaldi, E. Lucena-Sánchez, and G. Sciavicco. Temporal aspects of chlorophyll-a presence prediction around galapagos islands. In *International Conference on Technologies and Innovation*, pages 98–110. Springer, 2020.

[4] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, (3):1157–1182, 2003.

[5] H. Liu and H. Motoda. *Instance Selection and Construction for Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[6] S. Shirish and A. Ghatol. Use of instance typicality for efficient detection of outliers with neural network classifiers. In *Proc. of the 9th International Conference on Information Technology*, pages 225–228, 2006.

[7] Weizeng Ni. *A review and comparative study on univariate feature selection techniques*. PhD thesis, University of Cincinnati, 2012.

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

[9] H. Vafaie and K. De Jong. Genetic algorithms as a tool for feature selection in machine learning. In *Proc. of the 4th Conference on Tools with Artificial Intelligence*, pages 200–203, 1992.

[10] J. Yang and V. Honavar. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems and their Applications*, 13(2):44–49, 1998.

[11] A. Khalili. An overview of the new feature selection methods in finite mixture of regression models. *Journal of The Iranian Statistical Society*, 10(2):201–235, 2011.

[12] J.P. Barddal, F. Enembreck, H. Murilo-Gomes, A. Bifet, and B. Pfahringer. Boosting decision stumps for dynamic feature selection on data streams. *Information Systems*, 83:13–29, 2019.

[13] M. Hall. Time series analysis and forecasting with WEKA, 2014. Last accessed: May, 2019.

[14] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[15] G.H. John. Robust decision trees: Removing outliers from databases. In *Proc. of the 1st International Conference on Knowledge Discovery and Data Mining*, pages 174–179, 1995.

[16] D.B. Skalak and E.L. Rissland. Inductive learning in a mixed paradigm setting. In *Proc. of the 8th National Conference on Artificial Intelligence*, pages 840–847, 1990.

[17] S. Hawkins, H. He, G.J. Williams, and A.R. Baxter. Outlier detection using replicator neural networks. In *Proc. of the 4th International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180, 2002.

[18] N. Gornitz, M. Kloft, K. Rieck, and U. Brefeld. Toward supervised anomaly detection. *Journal of Artificial Intelligence Research*, 46:235– 262, 2013.

[19] H.D. Kuna, R. García-Martinez, and F.R. Villatoro. Outlier detection in audit logs for application systems. *Information Systems*, 44:22–33, 2014.

[20] M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Records*, 29(2):93–104, 2000.

[21] E.M. Knorr, R.T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *The Very Large Database Journal*, 8(3-4):237–253, 2000.

[22] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Records*, 29(2):427–438, 2000.

[23] S. Hadi and J. Simonoff. A stepwise procedure for identifying multiple outliers in linear regression. In *Proc. of the Meeting of the American Statistical Association Section on Statistical Computing*, pages 137–142, 1990.

[24] U. López, L. Trujillo, Y. Martínez, P. Legrand, E. Naredo, and S. Silva. RANSAC-GP: dealing with outliers in symbolic regression with genetic programming. In *Proc. of the 20th European Conference on Genetic Programming*, volume 10196 of *Lecture Notes in Computer Science*, pages 114–130, 2017.

[25] U. Lopez, L. Trujillo, and P. Legrand. Filtering outliers in one step with genetic programming. In *Proc. of the 15th International Conference on Parallel Problem Solving from Nature*, volume 11101 of *Lecture Notes in Computer Science*, pages 209–222, 2018.

[26] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui. A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition*, 74:406–421, 2017.

[27] R. Maronna, D. Martin, and V. Yohai. *Robust Statistics: Theory and Methods*. Wiley, 2006.

[28] R. Nunkesser and O. Morell. An evolutionary algorithm for robust regression. *Computational Statistics and Data Analysis*, 54(12):3242 – 3248, 2010.

[29] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[30] J. C. Riquelme, J. S. Aguilar-Ruiz, and M. Toro. Finding representative patterns with ordered projections. *Pattern Recognition*, 36(4):1009 –1018, 2003.

[31] A. Lumini and L. Nanni. A clustering method for automatic biometric template selection. *Pattern Recognition*, 3, 03 2006.

[32] R. Paredes and E. Vidal. Weighting prototypes - a new editing approach. In *Pattern Recognition, International Conference on*, volume 2, pages 25,26,27,28, Los Alamitos, CA, USA, sep 2000. IEEE Computer Society.

[33] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad. Prototype selection via prototype relevance. In J. Ruiz-Shulcloper and W. G. Kropatsch, editors, *Progress in Pattern Recognition, Image Analysis and Applications*, pages 153–160, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[34] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13(1):21–27, September 2006.

[35] D. B. Skalak. *Prototype Selection for Composite Nearest Neighbor Classifiers*. PhD thesis, University of Massachusetts, Amherst, MA, USA, 1997. UMI Order No. GAX97-37585.

[36] M. Elkano, M. Galar, J. Sanz, and H. Bustince. Chi-pg: A fast prototype generation algorithm for big data classification problems. *Neurocomputing*, 287:22–33, 2018.

[37] P. Hart. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theor.*, 14(3):515–516, September 2006.

[38] W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 01 1991.

[39] D. L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Systems, Man, and Cybernetics*, 2(3):408–421, 1972.

[40] D. R. Wilson and T. R. Martínez. Reduction techniques for instance-based learning algorithms. *Mach. Learn.*, 38(3):257–286, March 2000.

[41] Y. Li, Z. Hu, Y. Cai, and W. Zhang. Support vector based prototype selection method for nearest neighbor rules. In L. Wang, K. Chen, and Y. S. Ong, editors, *Advances in Natural Computation*, pages 528–535, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[42] J. A. Olvera-López, J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad. Sequential search for decremental edition. In M. Gallagher, J. P. Hogan, and F. Maire, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2005*, pages 280–285, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[43] I. Triguero, J. Derrac, S. García, and F. Herrera. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42:86 – 100, 02 2012.

[44] L. Yang, Q. Zhu, J. Huang, Q. Wu, D. Cheng, and X. Hong. Constraint nearest neighbor for instance reduction. *Soft Computing*, 23(24):13235–13245, Dec 2019.

[45] A. Arnaiz-González, J. F. Díez-Pastor, J. J. Rodríguez, and C. I. García-Osorio. Instance selection for regression by discretization. *Expert Syst. Appl.*, 54(C):340–350, July 2016.

[46] A. Arnaiz-González, J. F. Díez-Pastor, J. J. Rodríguez Díez, and C. I. García-Osorio. Instance selection for regression: Adapting drop. *Neurocomputing*, 201:66–81, 2016.

[47] A. Arnaiz-González, M. Blachnik, M. Kordos, and C. García-Osorio. Fusion of instance selection methods in regression task. *Information Fusion*, 30:69–79, 2016.

[48] J. A. Olvera-López, J. A. Carrasco-Ochoa, J. F. Martínez-Trinidad, and J. Kittler. A review of instance selection methods. *Artificial Intelligence Review*, 34(2):133–143, Aug 2010.

[49] N. Jankowski and M. Grochowski. Comparison of instances seletion algorithms i. algorithms survey. In L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, and L. A. Zadeh, editors, *Artificial Intelligence and Soft Computing - ICAISC 2004*, pages 598–603, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[50] M. Grochowski and N. Jankowski. Comparison of instance selection algorithms ii. results and comments. In L. Rutkowski, J. H. Siekmann, R. Tadeusiewicz, and L. A. Zadeh, editors, *Artificial Intelligence and Soft Computing - ICAISC 2004*, pages 580–585, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[51] J. Hoeting, A.E. Raftery, and D. Madigan. A method for simultaneous variable selection and outlier identification in linear regression. *Computational Statistics and Data Analysis*, 22(3), 1996.

[52] R.S. Menjoge and R.E. Welsch. A diagnostic method for simultaneous feature selection and outlier identification in linear regression. *Computational Statistics and Data Analysis*, 54(12):3181–3193, 2010.

[53] S.S. Kim, H.S. Park, and J.W. Krzanowski. Simultaneous variable selection and outlier identification in linear regression using the mean-shift outlier model. *Journal of Applied Statistics*, 35:283–291, 03 2008.

[54] A. Solanas, E. Romero, S. Gomez, J.M. Sopena, R. Alquezar, and J. Domingo-Ferrer. Feature selection and outliers detection with genetic algorithms and neural networks. In *Proc. of the 8th International Conference on Artificial Intelligence Research and Development*, volume 131, pages 41–48, 2005.

[55] J.T. de Souza, R.A. do Carmo, and G.A. de Lima. On the combination of feature and instance selection. In *Machine Learning*, pages 157–172. 2010.

[56] D. Fragoudis, D. Meretakis, and S. Likothanassis. Integrating feature and instance selection for text classification. In *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 501–506, 2002.

[57] C.F. Tsai, W. Eberle, and C.Y. Chu. Genetic algorithms in feature and instance selection. *Knowledge-Based Systems*, 39:240–247, 2013.

[58] K. Yu, X. Xu, M. Ester, and H.P. Kriegel. Feature weighting and instance selection for collaborative filtering: An information-theoretic approach. *Knowledge and Information Systems*, 5(2):201–224, 2003.

[59] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

[60] L. Breiman. Random forests. *Machine Learning*, 1(45):5–32, 2001.

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

[62] J.R. Quinlan. Simplifying decision trees. *International Journal of Human-Computer Studies*, 51(2):497–510, 1999.

[63] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[64] D.W. Hosmer and S. Lemeshow. *Applied Logistic Regression (2nd ed.)*. Wiley, 2000.

[65] S.S. Haykin. *Neural networks: A comprehensive foundation*. Prentice Hall, 1999.

[66] Junzhong Ji, Ning Zhang, Chunnian Liu, and Ning Zhong. An ant colony optimization algorithm for learning classification rules. In *Proc. of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 1034–1037, 2006.

[67] Y. Liu, Q. Zheng, Z. Shi, and J. Chen. Rule discovery with particle swarm optimization. In *Proc. of the 2004 Conference on Advanced Workshop on Content Computing (AWCC)*, volume 3309 of *Lecture Notes in Computer Science*, pages 291–296, 2004.

[68] M. Muntean, C. Rotar, I. Ileană, and H. Vălean. Learning classification rules with genetic algorithm. In *Proc. of the 8th International Conference on Communications*, pages 213–216, 2010.

[69] I.H. Witten, E. Frank, and M.A. Hall. *Data mining: practical machine learning tools and techniques, 3rd Edition*. Morgan Kaufmann, Elsevier, 2011.

[70] S. R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[71] S. K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data mining and knowledge discovery*, 2(4):345–389, 1998.

[72] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[73] J. R. Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[74] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Routledge, 2017.

[75] S. Shanmuganathan. Artificial neural network modelling: An introduction. In *Artificial neural network modelling*, pages 1–14. Springer, 2016.

[76] D. R. Hush and B. G. Horne. Progress in supervised neural networks. *IEEE signal processing magazine*, 10(1):8–39, 1993.

[77] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

[78] T. Hastie, R. Tibshirani, and J. Friedman. Unsupervised learning. In *The elements of statistical learning*, pages 485–585. Springer, 2009.

[79] S. Suthaharan. Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, 36:1–12, 2016.

[80] J.B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281 — 297, 1967.

[81] B.V. Dasarathy. *Nearest Neighbour (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, 1991.

[82] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester: John Wiley & Sons, 2001.

[83] F. Glover. Tabu search—part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[84] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, pages 1942–1948. IEEE, 1995.

[85] H. F. Wedde, M. Farooq, and Y. Zhang. Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior. In *International Workshop on Ant Colony Optimization and Swarm Intelligence*, pages 83–94. Springer, 2004.

[86] D. Simon. *Evolutionary optimization algorithms.* John Wiley & Sons, 2013.

[87] Y. Collette and P. Siarry. *Multiobjective Optimization: Principles and Case Studies.* Springer Berlin Heidelberg, 2004.

[88] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 2001.

[89] C. M. Fonseca, P. J. Fleming, et al. Genetic algorithms for multiobjective optimization: Formulationdiscussion and generalization. In *Icga*, volume 93, pages 416–423. Citeseer, 1993.

[90] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*, pages 82–87. Ieee, 1994.

[91] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation*, 2(3):221–248, 1994.

[92] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[93] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[94] J.J. Durillo and A.J. Nebro. jmetal: a java framework for multi-objective optimization. *Avances in Engineering Software*, 42:760 – 771, 2011.

[95] G. Medici, L. West, P. Chapman, and S. Banwart. Prediction of contaminant transport in fractured carbonate aquifer-types; case study of the permian magnesian limestone group (NE England, UK). *Environmental Science and Pollution Research*, 26(24):863–884, 2019.

[96] L. Wang, M.E. Stuart, M.A. Lewis, R.S. Ward, D. Skirvin, P.S. Naden, A.L. Collins, and M.J. Ascott. The changing trend in nitrate concentrations in major aquifers due to historical nitrate loading from agricultural land across england and wales from 1925 to 2150. *Science of the Total Environment*, 542:694–705, 2016.

[97] V. Re, C.H. Maldaner, J.J. Gurdak, M. Leblanc, T.C. Resende, and T.Y. Stigter. Topical collection: Climate-change research by early-career hydrogeologists. *Hydrogeology Journal*, 26(3):673–676, 2018.

[98] G. Martinelli, A. Minissale, and C. Verrucchi. Geochemistry of heavily exploited aquifers in the Emilia-Romagna region (Po Valley, Northern Italy). *Environmental Geology*, 4-4(36):195–206, 1998.

[99] G. Cremonini and F. Ricci Lucchi. *Guida alla Geologia del margine appenninico-padano (in Italian).* Pitagora Tecnoprint, 1982.

[100] G.M. Zuppi and E. Sacchi. Hydrogeology as a climate recorder: Sahara–Sahel (North Africa) and the po plain (Northern Italy). *Global and Planetary Change*, 40:79–91, 2004.

[101] A. Di Roma, E. Lucena-Sánchez, G. Sciavicco, and C. Vaccaro. An intelligent clustering method for devising the geochemical fingerprint of underground aquifers. *Heliyon*, 7(5):e07017, 2021.

[102] F. Jiménez, E. Lucena-Sánchez, G. Sánchez, and G. Sciavicco. Multi-objective evolutionary simultaneous feature selection and outlier detection for regression. *IEEE Access*, 9:135675–135688, 2021.

[103] A. Di Roma, E. Lucena-Sánchez, G. Sciavicco, and C. Vaccaro. Towards automatic fingerprinting of groundwater aquifers. In *International Conference on Technologies and Innovation*, pages 73–84. Springer, 2020.

[104] B. Jiang and J. Pei. Outlier detection on uncertain data: Objects, instances, and inferences. In *Proc. of the 27th International Conference on Data Engineering*, pages 422–433, 2011.

[105] G. Krishnaì and R. Vadlamani. Outlier detection using evolutionary computing. In *Proc. of the 1st International Conference on Informatics and Analytic*, pages 1–6, 2016.

[106] L. Martí, A. Fansi-Tchango, L. Navarro, and M. Schoenauer. Progressively adding objectives: A case study in anomaly detection. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 593–600, 2017.

[107] Z. Chen, K.Y. Chai, B. Sung Lee, L. Chiew Tong, and Y. Jin. Evolutionary multi-objective optimization based ensemble autoencoders for image outlier detection. *Neurocomputing*, 309:192–200, 2018.

[108] B.P. Li, A. Greig, J.X. Zhao, K.D. Collerson, K.S. Quan, Y.H. Meng, and Z.L. Ma. Icp-ms trace element analysis of song dynasty porcelains from Ding, Jiexiu and Guantai kilns, north China. *Journal of Archaeological Sciences*, 32:251–259, 2005.

[109] J. Ross, A.L. Jaques, J. Perguson, D.H. Green, S.Y. O'Reilly, R.V. Danchin, and A.J.A. Janse. Sodium in garnet and potassium in clinopyroxene: criteria for classifying mantle eclogites. *Kimberlites and Related Rocks*, pages 27–832, 1989.

[110] S. Galleta, B.M. Jahn, B. Van Vliet Lanoë, A. Dia, and E. Rossello. Loess geochemistry and its implications for particle origin and composition of the upper continental crust. *Earth Planet Science Letters*, pages 157–172, 1989.

[111] A. Ranjbar, N. Mahjouri, and C. Cherubini. Development of an efficient conjunctive meta-model-based decision-making framework for saltwater intrusion management in coastal aquifers. *Journal of Hydro-environment Research*, 2019.

[112] M. Stuart, D. Lapworth, J. Thomas, and L. Edwards. Fingerprinting groundwater pollution by microorganic compounds in catchments with contrasting contaminant sources. *The Science of the Total Environment*, 468-469C:564–577, 09 2013.

[113] M. Balseiro-Romero, F. Macías, and C. Monterroso. Characterization and fingerprinting of soil and groundwater contamination sources around a fuel distribution station in galicia (NW spain). *Environmental Monitoring and Assessment*, 188:1–15, 2016.

[114] P.C. Nathaniel and L.K. Lautz. Discriminant analysis as a decision-making tool for geochemically fingerprinting sources of groundwater salinity. *Science of The Total Environment*, 618:379 – 387, 2018.

[115] S. Pepi and C. Vaccaro. Geochemical fingerprints of "'Prosecco" wine based on major and trace elements. *Environmental Geochemestry and Health*, 2(40):833–847, 2018.

[116] J.W. Morse and R.S. Arvidson. The dissolution kinetics of major sedimentary carbonate minerals. *Earth-Science Reviews*, 58:51–84, 2002.

[117] Z. Zhihao, X. Changlai, Y. Weifei, A. Oluwafemi, and L. Xiujuan. Source and mobilization mechanism of iron, manganese and arsenic in groundwater of shuangliao city, northeast china. *Water*, 12:534–551, 2020.

[118] C. Li nan Baena, B. Andreo, J. Mudry, and F. Carrasco-Cantos. Groundwater temperature and electrical conductivity as tools to characterize flow patterns in carbonate aquifers: The Sierra de las Nieves karst aquifer, southern Spain. *Hydrogeology Journal*, 17:843 — 853, 2009.

[119] A.N. Tiwari, V.P. Nawale, J.A. Tambe, and Y. Satyakumar. Variation in calcium and magnesium ratio with increasing electrical conductivity of groundwater from shallow basaltic aquifers of Maharashtra (India). *Journal of Environmental Science and Engineering*, 52:311 – 314, 2010.

[120] B. Capaccioni, M. Didero, C. Paletta, and L. Didero. Saline intrusion and refreshening in a multilayer coastal aquifer in the Catania Plain. *Journal of Hydrology*, 307:1 – 16, 2005.

[121] D. Schwela. Air pollution and health in urban areas. *Reviews on environmental health*, 15(1-2):13–42, 2000.

[122] R. Beelen, O. Raaschou-Nielsen, M. Stafoggia, Z. J. Andersen, G. Weinmayr, B. Hoffmann, K. Wolf, E. Samoli, P. Fischer, M. Nieuwenhuijsen, P. Vineis, W. W. Xun, K. Katsouyanni, K. Dimakopoulou, A. Oudin, B. Forsberg, L. Modig, A. S. Havulinna, T. Lanki, A. Turunen, B. Oftedal, W. Nystad, P. Nafstad, U. De Faire, N. L. Pedersen, C. . Östenson, L. Fratiglioni, J. Penell, M. Korek, G. Pershagen, K. T. Eriksen, K. Overvad, T. Ellermann, M. Eeftens, P. H. Peeters, K. Meliefste, M. Wang, B. Bueno-De-Mesquita, D. Sugiri, U. Krämer, J. Heinrich, K. De Hoogh, T. Key, A. Peters, R. Hampel, H. Concin, G. Nagel, A. Ineichen, E. Schaffner, N. Probst-Hensch, N. Künzli, C. Schindler, T. Schikowski, M. Adam, H. Phuleria, A. Vilier, F. Clavel-Chapelon, C. Declercq, S. Grioni, V. Krogh, M. . Tsai, F. Ricceri, C. Sacerdote, C. Galassi, E. Migliore, A. Ranzi, G. Cesaroni, C. Badaloni, F. Forastiere, I. Tamayo, P. Amiano, M. Dorronsoro, M. Katsoulis, A. Trichopoulou, B. Brunekreef, and G. Hoek. Effects of long-term exposure to air pollution on natural-cause mortality: An analysis of 22 European cohorts within the multicentre ESCAPE project. *The Lancet*, 383(9919):785–795, 2014.

[123] M. Kowalska, M. Skrzypek, M. Kowalski, J. Cyrys, N. Ewa, and E. Czech. The relationship between daily concentration of fine particulate matter in ambient air and exacerbation of respiratory diseases in Silesian agglomeration, Poland. *International Journal of Environmental Research and Public Health*, 16(7), 2019.

[124] P. Holnicki, M. Tainio, A. Kałuszko, and Z. Nahorski. Burden of mortality and disease attributable to multiple air pollutants in Warsaw, Poland. *International Journal of Environmental Research and Public Health*, 14(11), 2017.

[125] J. Schwartz. Lung function and chronic exposure to air pollution: A cross-sectional analysis of NHANES II. *Environmental research*, 50(2):309–321, 1989.

[126] G. Cesaroni, C. Badaloni, C. Gariazzo, M. Stafoggia, R. Sozzi, M. Davoli, and F. Forastiere. Long-term exposure to urban air pollution and mortality in a cohort of more than a million adults in Rome. *Environmental health perspectives*, 121(3):324–331, 2013.

[127] R. D. Peng, F. Dominici, and T. A. Louis. Model choice in time series studies of air pollution and mortality. *Journal of the Royal Statistical Society. Series A: Statistics in Society*, 169(2):179–203, 2006.

[128] J. Schwartz. Air pollution and blood markers of cardiovascular risk. *Environmental Health Perspectives*, 109(SUPPL. 3):405–409, 2001.

[129] T. F. Mar, G. A. Norris, J. Q. Koenig, and T. V. Larson. Associations between air pollution and mortality in Phoenix, 1995-1997. *Environmental health perspectives*, 108(4):347–353, 2000.

[130] J. K. Vanos, S. Cakmak, L. S. Kalkstein, and A. Yagouti. Association of weather and air pollution interactions on daily mortality in 12 Canadian cities. *Air Quality, Atmosphere and Health*, 8(3):307–320, 2015.

[131] Luke D. Knibbs, Adriana M. Cortés de Waterman, Brett G. Toelle, Yuming Guo, Lyn Denison, Bin Jalaludin, Guy B. Marks, and Gail M. Williams. The australian child health and air pollution study (ACHAPS): A national population-based cross-sectional study of long-term exposure to outdoor air pollution, asthma, and lung function. *Environment International*, 120:394 – 403, 2018.

[132] L. A. Cifuentes, J. Vega, K. Köpfer, and L. B. Lave. Effect of the fine fraction of particulate matter versus the coarse mass and other pollutants on daily mortality in Santiago, Chile. *Journal of the Air and Waste Management Association*, 50(8):1287–1298, 2000.

[133] J. A. Kamińska, F. Jiménez, E. Lucena-Sánchez, G. Sciavicco, and T. Turek. Lag variables in nitrogen oxide concentration modelling: A case study in wrocław, poland. *Atmosphere*, 11(12):1293, 2020.

[134] E. Lucena-Sánchez, F. Jiménez, G. Sciavicco, and J. Kamińska. Simple versus composed temporal lag regression with feature selection, with an application to air quality modeling. In *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, pages 1–8. IEEE, 2020.

[135] J. A. Kamińska, G. Sciavicco, E. Lucena-Sánchez, and F. Jiménez. Lag variables in air pollution modeling based on traffic flow and meteorological factors. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 51, page 1, 2020.

[136] F. Jiménez, J. Kaminska, E. Lucena Sanchez, J. Palma, G. Sciavicco, et al. Multi-objective evolutionary optimization for time series lag regression. In *6th International Conference on Time Series and Forecasting*, pages 373–384. Godel Impresiones Digitales, 2019.

[137] J. Kamińska, E. Lucena-Sánchez, and G. Sciavicco. Temporal aspects in air quality modeling—a case study in wrocław. *Air, Soil and Water Research*, 13:1178622120975829, 2020.

[138] J.A. Kamińska. The use of random forests in modelling short-term air pollution effects based on traffic and meteorological conditions: A case study in Wrocław. *Journal of environmental management*, 217:164–174, 2018.

[139] J.A. Kamińska. A random forest partition model for predicting NO_2 concentrations from traffic flow and meteorological conditions. *Science of the Total Environment*, 651:475–483, 2019.

[140] E. Chianese, F. Camastra, A. Ciaramella, T. C. Landi, A. Staiano, and A. Riccio. Spatio-temporal learning in predicting ambient particulate matter concentration by multi-layer perceptron. *Ecological informatics*, 49:54–61, 2019.

[141] PJ G. Nieto, F. S. Lasheras, E. García-Gonzalo, and FJ de Cos Juez. Pm10 concentration forecasting in the metropolitan area of oviedo (Northern Spain) using models based on SVM, MLP, VARMA and ARIMA: A case study. *Science of The Total Environment*, 621:753–761, 2018.

[142] N.L. Gilbert, M.S. Goldberg, B. Beckerman, J.R. Brook, and M. Jerrett. Assessing spatial variability of ambient nitrogen dioxide in montreal, canada, with a land-use regression model. *Journal of the Air & Waste Management Association*, 55(8):1059–1063, 2005.

[143] S.B. Henderson, B. Beckerman, M. Jerrett, and M. Brauer. Application of land use regression to estimate long-term concentrations of traffic-related nitrogen oxides and fine particulate matter. *Environmental science & technology*, 41(7):2422–2428, 2007.

[144] G. Hoek, R. Beelen, K. De Hoogh, D. Vienneau, J. Gulliver, P. Fischer, and D. Briggs. A review of land-use regression models to assess spatial variation of outdoor air pollution. *Atmospheric environment*, 42(33):7561–7578, 2008.

[145] Y. Son, A.R. Osornio-Vargas, M.S. O'Neill, P. Hystad, J.L. Texcalac-Sangrador, P. Ohman-Strickland, Q. Meng, and S. Schwander. Land use regression models to assess air pollution exposure in mexico city using finer spatial and temporal input parameters. *Science of the Total Environment*, 639:40–48, 2018.

[146] M. Chalfen and J. Kamińska. Analiza wykorzystania roweru miejskiego we wrocławiu (in Polish). *Autobusy. Technika, Eksploatacja, Systemy Transportowe*, 17(6):543–545, 2016.

[147] A. T. Toback, J. S. Hearne, B. Kuritz, A. J. Marchese, and R. P. Hesketh. The effect of ambient temperature and humidity on measured idling emissions from diesel school buses. *SAE Technical Papers*, 2004.

[148] S.R. Krause, D.F. Merrion, and G.L. Green. Effect of inlet air humidity and temperature on diesel exhaust emissions. *SAE Technical Papers*, 1973.

[149] Paula López-Arce, Javier Garcia-Guinea, Mercedes Gracia, and Joaquín Obis. Bricks in historical buildings of toledo city: characterisation and restoration. *Materials Characterization*, 50(1):59–68, 2003.

[150] E. Marrocchino, G. Sciavicco, E. Lucena-Sánchez, and C. Vaccaro. On intelligent fingerprinting of antique buildings from clay composition. In *International Conference on Technologies and Innovation*, pages 33–47. Springer, 2021.

[151] L. Belkhiri, L. Mouni, T. Sheikhy Narany, and A. Tiri. Evaluation of potential health risk of heavy metals in groundwater using the integration of indicator kriging and multivariate statistical methods. *Groundwater for Sustainable Development*, 4:12 – 22, 2017.

[152] I. Kozyatnyk, L. Lövgren, M. Tysklind, and P. Haglund. Multivariate assessment of barriers materials for treatment of complex groundwater rich in dissolved organic matter and organic and inorganic contaminants. *Journal of Environmental Chemical Engineering*, 5(4):3075 – 3082, 2017.

[153] C.K. Singh, A. Kumar, S. Shashtri, A. Kumar, P. Kumar, and J. Mallick. Multivariate statistical analysis and geochemical modeling for geochemical assessment of groundwater of Delhi, India. *Journal of Geochemical Exploration*, 175:59 – 71, 2017.

[154] A. Ozdemir. Gis-based groundwater spring potential mapping in the sultan mountains (Konya, Turkey) using frequency ratio, weights of evidence and logistic regression methods and their comparison. *Journal of Hydrology*, 411(3):290 – 308, 2011.

[155] L. Pizzol, A. Zabeo, A. Critto, E. Giubilato, and A. Marcomini. Risk-based prioritization methodology for the classification of groundwater pollution sources. *Science of The Total Environment*, 506:505 – 517, 2015.

[156] A. Mair and A.I. El-Kadi. Logistic regression modeling to assess groundwater vulnerability to contamination in hawaii, usa. *Journal of Contaminant Hydrology*, 153:1 – 23, 2013.

[157] A. Menció, J. Mas-Pla, N. Otero, O. Regàs, M. Boy-Roura, R. Puig, J. Bach, C. Domènech, M. Zamorano, D. Brusi, and A. Folch. Nitrate pollution of groundwater; all right..., but nothing else? *Science of The Total Environment*, 539:241 – 251, 2016.

[158] H. Farhadian and H. Katibeh. New empirical model to evaluate groundwater flow into circular tunnel using multiple regression analysis. *International Journal of Mining Science and Technology*, 27(3):415 – 421, 2017.