

# Three-objective constrained evolutionary instance selection for classification: Wrapper and filter approaches

Fernando Jiménez<sup>a,\*</sup>, Gracia Sánchez<sup>a</sup>, José Palma<sup>a</sup>, Guido Sciacvico<sup>b</sup>

<sup>a</sup> Department of Information and Communication Engineering, University of Murcia, Spain

<sup>b</sup> Department of Mathematics and Computer Science, University of Ferrara, Italy

## ARTICLE INFO

### Keywords:

Multi-objective evolutionary algorithms  
Multi-objective constrained optimization  
Instance selection  
Classification

## ABSTRACT

The large amount of data that is produced today with new technologies is an impediment for machine learning algorithms to work correctly, both due to the memory requirements and the necessary execution times. That is why the processes of reducing both the quantity and the size of the data are increasingly important. One of these processes is the so-called *instance selection*. In this paper we propose three-objective constrained optimization models to formulate instance selection wrapper and filter methods (separately) for classification problems, which are solved with multi-objective evolutionary algorithms and multi-objective differential evolution. In the proposed instance selection wrapper method, an objective is added to the usual ones to minimize the generalization error of the classifier. The proposed instance selection filter method simultaneously optimizes the correlation, redundancy and consistency of the datasets. Instance retention constraints are imposed on optimization models to retain a maximum percentage of samples, established by the decision maker, in big data scenarios. The experiments have been designed to compare (1) the *NSGA-II* and *MODE* algorithms, (2) two- and three-objective optimization models, (3) two different constraint handling techniques, and (4) the proposed evolutionary approaches and other 12 non-evolutionary approaches used in literature. The proposed wrapper and filter instance selection methods have been used in a real-world business engineering application, and have also been validated using three public datasets to facilitate the replicability of the research results. The results of the experiments show the superiority of the three-objective constrained evolutionary techniques proposed in this paper over the non-evolutionary techniques and over the two-objective evolutionary approaches used in the literature.

## 1. Introduction

*Instance selection* is defined in Liu and Motoda (2001) as the process that selects or searches for a representative portion of data that can fulfill a KDD task as if the whole data is used. Instance selection reduces an original dataset to a manageable volume, which leads to a reduction in the computational resources needed to complete the learning process. Instance selection is part of the *data reduction* process within the *data pre-processing* step in *data mining* (Fayyad et al., 1996).

As in *feature selection* (Liu and Motoda, 1998), instance selection methods can be divided into *filter* and *wrapper* methods. Instance selection filter methods use information measures or rules to evaluate instance subsets, such as *weakness* (Riquelme et al., 2003a), *clustering* (Lumini and Nanni, 2006), *weighting* (Paredes and Vidal, 2000), *relevance* (Olvera-López et al., 2008), etc. Instance selection wrapper methods use a classifier to evaluate candidate instance subsets. Most of the instance selection wrapper methods have been proposed based on the  $k - NN$  classifier (Cover and Hart, 2006), and this type of instance

selection is called *prototype selection* (Skalak, 1997). Prototype selection methods attempt to select a representative subset of samples from the training set, and *prototype generation* methods (Elkano et al., 2018) generate a small set of artificial prototypes to replace the original training set. These methods are usually based on *misclassification* (Hart, 2006; Aha et al., 1991; Wilson, 1972), although there are others based on *associate sets* (Wilson and Martínez, 2000) or *support vector machines* (Li et al., 2005). As feature selection, instance selection can be considered as a search problem, and *forward*, *backward* and *mixed* search strategies can also be applied for instance selection (Olvera-López et al., 2005), besides metaheuristics such as *Evolutionary Algorithms* (Cano et al., 2004), *Artificial Immune Systems* (Garain, 2008) or *Tabu Search* (Lleó and Ferri, 2001). Finally, instance selection methods can also be categorized according to the *selection strategy* (Triguero et al., 2012) as *condensation algorithms*, that attempt to remove the instances far away from the decision surface, *edition algorithms*, that removes noises to improve the classification accuracy, and *hybrid algorithms*, that remove both central and border instances, and a reduced set can be

\* Corresponding author.

E-mail address: [fernand@um.es](mailto:fernand@um.es) (F. Jiménez).

obtained by merging border and core instances (Yang et al., 2019). Instance selection is not only required for classification tasks, but also for regression (Arnaiz-González et al., 2016c,b,a). An overview of instance selection methods can be found in Olvera-López et al. (2010), Jankowski and Grochowski (2004), Grochowski and Jankowski (2004b).

A drawback of the  $k - NN$  algorithm is that it requires storing the entire set of instances in memory. This is an impediment in practice when handling large amounts of data (Brighton and Mellish, 2002). In these cases, specific approaches for *Big Data* should be used (Arnaiz-González et al., 2016d; Cano et al., 2005; García-Osorio et al., 2010), or use another type of machine learning algorithm more efficient and appropriate for the required learning task, such as *decision trees* (Dean Bennette, 2011). Generalization performance of a classifier and the proportion of data reduction are factors in conflict because, in general, maximizing dataset reduction implies decreasing accuracy and vice versa. Therefore, this problem can be naturally approached with *multi-objective optimization* (Deb, 2005). *Multi-objective evolutionary algorithms* (MOEAs) (Deb, 2001; Coello et al., 2002) are particularly appropriate for multi-objective optimization since they can capture a set of non-dominated solutions for the problem in a single run of the algorithm. The main works that guide instance selection as a multi-objective problem through evolutionary algorithms aggregate the objective functions into a single objective function which to optimize. In addition, these works focus on prototype selection using  $k - NN$  classifiers. There are very few works that perform a Pareto-based multi-objective evolutionary search, and those are also focusing on prototype selection. On the other hand, the existing works for instance selection focus the problem on reducing the set of instances without taking into account the maximum number of these. In practice, this is an inconvenience in presence of a large amount of data, where the selection of instances continues involving a large amount of data that prohibits classification task from being performed at reasonable time. Another issue that is not addressed by many of the existing works is the *overfitting prevention* of the classifier over the reduced data set, with the consequent loss of prediction capability. Most of the data reduction works based on Pareto-based MOEAs deal with classification problems and those dedicated to regression tasks are very scarce and, again, use  $k - NN$  as learning algorithm (Kordos and Kapa, 2018; Kordos et al., 2019).

This paper focuses on instance selection for classification problems. We propose novel multi-objective constrained optimization models for instance selection, which are solved by MOEAs. In summary, we can list the following contributions regarding related works:

1. We propose a novel three-objective constrained boolean optimization model for wrapper instance selection. We implement overfitting prevention by adding an objective for instance subset performance maximization in *p-fold cross-validation* evaluation mode. We use a generic approach that includes any type of machine learning algorithm (not restricted to  $k - NN$ ). We compared  $k - NN$ -based approach with approaches based on others classifiers, such as *C4.5* and *Random Forest*, using *accuracy* and *retention percentage* performance metrics. In addition, we propose a novel three-objective constrained boolean optimization model for filter instance selection in classification problems based on *consistency* and *correlation* measures. A decision making process based on *goal programming* and *compromise programming* has been proposed to choose a non-dominated solution from the three-objective Pareto front. This decision-making process is also used as a performance metric for comparison of instance selection methods.
2. Our proposal allows to limit the reduced instance set cardinality by adding an user-defined constraint into the optimization problem. We compared two different approaches for constraint handling, based on (1) repair algorithms and (2) evolution of infeasible solutions toward feasibility using *Min-Max* criterion.

3. *NSGA-II* and *MODE* have been implemented in the *Weka* machine learning platform to solve the proposed instance selection methods. These implementations use self-adaptive variation operators. The software will soon be officially published on the *Weka* platform for public use, thus allowing the replicability and comparison of results. Three public datasets of the *UCI Machine Learning Repository* have been used, also for replicability reasons.
4. We compared our three-objective constrained approach with the two-objective approach used in literature, as well as with non-evolutionary instance selection methods implemented in open source public software. We also apply our approach to a real-life classification problem in the business engineering area, which is analyzed from the perspective of the *small data* paradigm. We have proposed the *retained-removed sample proximity* metric to compare instance selection methods.

In summary, this work skilfully combines several concepts, approaches, techniques and components, such as multi-objective evolutionary algorithms, multi-objective constrained optimization, instance selection, classification, feature selection, filter structure and wrapper structure. This is a typical combination novelty, This is a typical combination novelty, which can be used in joint big data and machine learning scenarios.

The paper has been organized as follows: Section 2 is a brief introduction to multi-objective constrained optimization, describes three MOEAs widely used in literature and shows the state-of-the-art of MOEAs for instance selection; Section 3 presents a novel multi-objective constrained optimization model for instance selection; Section 4 describes the main components of the MOEA that solves the proposed optimization model for instance selection, including a decision making process for choosing the final non-dominated solution; Section 5 describes the datasets and the set of experiments performed in this paper, together with the analysis of obtained results; Section 7 analyzes a real-life case study with data obtained from a contact center in a city in Italy; finally, Section 8 presents our conclusions and outlines future work.

## 2. Background

### 2.1. Multi-objective constrained optimization

Instance selection can be framed as an *optimization problem* (Huibertus et al., 2004), and expressed in terms of *mathematical programming* (Sinha, 2006; Karloff, 1991; Bertsekas, 1999). We are interested in a class of mathematical programming problems called *multi-objective constrained optimization problems* (Collette and Siarry, 2004), which can be formally defined, for  $l$  objectives and  $m$  constraints, as follows:

$$\begin{aligned} \text{Min./Max.} \quad & f_i(\mathbf{x}), \quad i = 1, \dots, l \\ \text{subject to} \quad & g_j(\mathbf{x}) \leq 0, \quad j = 1, \dots, m \end{aligned} \quad (1)$$

where  $f_i(\mathbf{x})$  (called *objectives*) and  $g_j(\mathbf{x})$  (called *constraints*) are arbitrary functions. Optimization problems can be naturally separated into two categories: those with discrete variables, called *combinatorial optimization problems*, and those with continuous variables. In combinatorial optimization problems, we are looking for objects from a finite, or countably infinite, set  $\mathcal{X}$ , typically integers, sets, permutations, or graphs. In problems with continuous variables, instead, we look for real parameters belonging to some continuous domain. In optimization problem (1),  $\mathbf{x} = \{x_1, x_2, \dots, x_w\} \in \mathcal{X}^w$  represents the set of *decision variables*, where  $\mathcal{X}$  is the domain for each variable  $x_k$ ,  $k = 1, \dots, w$ .

Let  $F = \{\mathbf{x} \in \mathcal{X}^w \mid g_j(\mathbf{x}) \leq 0, j = 1, \dots, m\}$  be the set of all *feasible solutions* to (1). We want to find a subset of solutions  $S \subseteq F$  called *non-dominated set* (or *Pareto optimal set*). A solution  $\mathbf{x} \in F$  is *non-dominated* if there is no other solution  $\mathbf{x}' \in F$  that dominates  $\mathbf{x}$ , and a solution  $\mathbf{x}'$  *dominates*  $\mathbf{x}$  if and only if there exists  $i$  ( $1 \leq i \leq l$ ) such that  $f_i(\mathbf{x}')$  improves  $f_i(\mathbf{x})$ , and for every  $i$  ( $1 \leq i \leq l$ ),  $f_i(\mathbf{x})$  does not improve

$f_i(\mathbf{x}')$ . In other words,  $\mathbf{x}'$  dominates  $\mathbf{x}$  if and only if  $\mathbf{x}'$  is better than  $\mathbf{x}$  for at least one objective, and not worse than  $\mathbf{x}$  for any other objective. For minimization problems, the set  $S$  of non dominated solutions of (1) can be formally defined as:

$$S = \{\mathbf{x} \in \mathcal{F} \mid \nexists \mathbf{x}' \in \mathcal{F} \mid D(\mathbf{x}', \mathbf{x})\}$$

where:

$$D(\mathbf{x}', \mathbf{x}) \equiv \exists i, 1 \leq i \leq l, f_i(\mathbf{x}') < f_i(\mathbf{x}) \wedge \forall i, 1 \leq i \leq l, f_i(\mathbf{x}') \leq f_i(\mathbf{x})$$

Once the set of optimal solutions is available, the most satisfactory one can be chosen by applying a preference criterion. A combinatorial optimization problem in which the objectives or constraints are arbitrary functions is, in general, intractable. In principle, any search algorithm can be used to solve non-linear combinatorial optimization problems, although they are not guaranteed to find an optimal solution. *Metaheuristics* methods widely established in the literature such as *evolutionary algorithms* (Jiménez and Verdegay, 2001), *tabu search* (Hou et al., 2020), *particle swarm optimization* (Haoran et al., 2020), or other more recently created methods such as *harris hawks optimization* (Heidari et al., 2019), *emperor penguins colony* (Dhiman and Kumar, 2018), *bat algorithm* (Luo et al., 2020), *salp swarm algorithm* (Dhiman, 2019), *whale optimization algorithm* (Mirjalili and Lewis, 2016), *biogeography-based optimization* (Chen et al., 2020), and many others, are typically used to find approximate solutions for complex optimization problems. We focus on MOEAs in this paper because they are well-established algorithms in the literature for multi-objective optimization, as well as because of our experience over the last decade. MOEAs are also suitable for large-scale multi-objective optimization (Tian et al., 2021), including multi-objective instance selection problems in big data sceneries.

## 2.2. Multi-objective evolutionary algorithms

This section describes *NSGA-II* and *MODE*. These algorithms are two of the most widely used MOEAs in the literature for multi-objective constrained optimization, and in this paper they have been implemented as search strategies for instance selection methods and have been compared in experiments.

### 2.2.1. NSGA-II

The *NSGA-II* (Deb, 2001) is a MOEA that uses  $(\mu + \lambda)$  elitist strategy (Algorithm 1) with  $\mu = \lambda = N$ , where  $\mu$  corresponds to the number of parents,  $\lambda$  refers to the number of children, and  $N$  is the population size. *NSGA-II* uses a *binary tournament selection* (Algorithm 2) and a rank function based on Pareto fronts and *crowding* (Algorithm 3 and Algorithm 4). The rank of an individual in a population is the non-domination level of the individual in the whole population. In turn, individuals with the same level of non-domination are sorted according to their degree of crowding at that level.<sup>1</sup> *NSGA-II* uses the *fast non-dominated sorting* (Deb et al., 2002) to assign a non-domination level to each individual. It compares each solution with the rest of the solutions and stores the results so as to avoid duplicate comparisons between every pair of solutions. *NSGA-II* was designed by K. Deb et al. and has proven to be a very powerful and fast algorithm in multi-objective

<sup>1</sup> *RankCrowding* (Algorithm 3) is an order relation where  $rank(S, I)$  and  $rank(S, J)$  return the non-domination level (number of front) of the individual  $I$  and  $J$  respectively in a set of individuals  $S$ . In problems with restrictions such as (1) the ranking of a feasible individual is always lower (which means better) than that of a unfeasible individual, and unfeasible individuals are assigned a ranking according to the function  $\max_j \{g_j(\mathbf{x})\}$  which is minimized. Another alternative is to handle the constraints with repair algorithms that are applied after initialization, crossover and mutation, with which only feasible individuals exist in the population.

optimization contexts of all kinds. Most researchers in multi-objective evolutionary computation use *NSGA-II* as a baseline to compare the performance of their own algorithms. Although *NSGA-II* was developed in 2002, it is still a state-of-the-art algorithm and it remains a challenge to overcome it. There is a recently updated improved version for *many-objective optimization* problems called *NSGA-III* (Deb and Jain, 2014).

### 2.2.2. MODE

*Differential Evolution* (DE) algorithm is an instance of an evolutionary algorithm originally developed Storn and Price (1995). DE generates new individuals based on the weighted difference between two randomly selected individuals of the population added to a third (base) individual. Together with the selection, this variation self-organizes the sampling of the problem space, bounding it to known areas of interest. DE constitutes a family of algorithms whose variants depend on the way the individuals are selected for computing mutation process, number of pairs of individuals to find the difference individual, and the type of crossover used. The general convention of the DE variant can be denoted as *DE/p/q/r*, where  $p$  represents the way of selecting the base individual for mutation (*rand* denotes a random individual, *best* denotes the best individual, and *rand-to-best* denotes an individual between a *random* individual and the *best* individual),  $q$  denotes the number of pairs of individuals to be selected for differential mutation (usually 1 or 2) and  $r$  denotes the type of crossover to be performed (*bin* denotes *Binomial* and *exp* denotes *Exponential*). In classical DE, all individuals in the population are always mutated with differential mutation and then crossover is performed between the parent and the mutated individual with  $CR$  probability. The fitness of the individual resulting from the crossover is compared to the fitness of the parent, and only the individual with better fitness goes to the next generation (selection), being therefore an elitist algorithm.

*Multi-objective DE* follows same steps of the classical DE except for some changes related to the optimization of more than one objective function. One way to approach *MODE* is by combining the characteristics of DE with those of *NSGA-II* (Xue et al., 2004), that is,  $(\mu + \lambda)$  elitist strategy with fast non-dominated sorting and crowding along with the differential mutation followed by the crossover of DE. That approach, named *MODE* in this paper, is shown in the Algorithm 5. The main differences between Algorithm 5 and Algorithm 1 are that Algorithm 5 does not contemplate binary tournament selection and performs the specific DE variation operators. In this paper we use the Algorithm 5 with the variant *DE/best/1/bin*.

## 2.3. Multi-objective evolutionary instance selection

In this section we review the work of other authors who have used Pareto-based MOEAs for instance selection, prototype selection or prototype generation. We do not include in this review those works that aggregate multiple objective functions for single-objective optimization.

A Pareto-based MOEA for simultaneous instance selection and feature selection is proposed in Chen et al. (2005). They use a three-objective optimization model where instance and feature sizes are minimized simultaneously by using two independent objectives, and a third objective maximizes the accuracy of an  $1 - NN$  classifier trained with the reduced dataset and evaluated in the full dataset. Authors propose *IMOEA* (Ho et al., 2004) to solve the optimization problem, which is compared with *IGA* (single-objective version of *IMOEA*), *SPEA* (Zitzler and Thiele, 2000) and *DROP5* (Wilson and Martínez, 2000) using 11 public datasets from the *UCI Machine Learning Repository* with hold-out validation.

A three-step approach for prototype generation is proposed in Rosales-Pérez et al. (2014a). The first step is to obtain a weight for each training instance related to their discrimination power, the second step is to generate an initial set of prototypes through the selection



of samples from the training set, and in the third step the positions of the prototypes are adjusted through an evolutionary process. A multi-objective optimization problem is solved where two objectives are considered: (1) minimization of the  $1 - NN$  classification error in the training set when using the prototypes; and (2) minimization of the relative reduction rate. A constraint is imposed to ensure that all possible sets of prototypes have at least one prototype per class, which is handled following a penalty function approach. Authors used the  $(1+1)$ -PAES (Knowles and Corne, 2000) for solving this problem over 59 datasets taken from the KEEL repository (Alcalá-Fdez et al., 2010) evaluated on 10-fold cross-validation and compared with  $1 - NN$ , AMPSO, GENN, LVQTC, MSE, PSCSA and PSO (Triguero et al., 2012). An extension of this approach for simultaneous instance selection and feature selection is proposed in Rosales-Pérez et al. (2014b) where a third objective is added for minimization of the number of features. The authors also propose EMOMIS-PbE (Rosales-Pérez et al., 2017), a two-objective MOEA to minimize the error rate of a *support vector machine* (SVM) and the reduction ratio. The authors apply ensembles of classifiers to combine the non-dominated solutions of the final population, using different techniques (GPE, IERE, CIE, MDE and Boosting). 43 datasets from (Alcalá-Fdez et al., 2010) are used in the experiments.

In Pighetti et al. (2015) a MOEA is proposed to optimize the training samples used by an SVM classifier in order to improve its learning rate. The MOEA considers two filter-type objectives. An objective identifies the worst and best individuals to determine the center of the class. The other objective is to find a diverse set of items and have the MOEA explore space. The method is evaluated for the classification of images of the database of standard images Caltech256 (30000 images distributed in 256 classes).

The proposed approach in Escalante et al. (2017) for prototype generation also uses two objectives functions for accuracy and reduction rate maximization. The main difference with respect to the works of other authors is that a dynamic fitness function is used. In each iteration of the MOEA a stratified validation set is randomly chosen from the full dataset which is used to evaluate the  $1 - NN$  classifiers trained with the reduced datasets. A constraint is also imposed to ensure that all possible sets of prototypes have at least one prototype per class. This constraint is handled in this case by means of *ad hoc* initialization and variation operators. The authors propose NSGA-II to solve the two-objective optimization problem, and they used 59 datasets from Alcalá-Fdez et al. (2010) in the experiments, evaluated on 10-fold cross-validation and compared with 25 methods considered in Triguero et al. (2012).

In Acampora et al. (2018) a MOEA with two objectives is proposed for instance selection that minimizes the error rate of an SVM and the reduction rate, which uses PESA-II (Corne et al., 2001) as optimizer. An asymmetric variation strategy is used to decrease the selected subset size, and a decision-making mechanism based on a sum model is also designed to choose a good solution from the Pareto front. 36 datasets from the UCI Machine Learning Repository are used in the experiments.

In Kordos and Kapa (2018) the authors use NSGA-II to solve a two-objective optimization problem for instance selection and *instance weighting* for regression, where the root mean square error of  $k - NN$  in the training set when using the reduced set of instances is minimized. The second objective is the minimization of number of selected instances. In order to evaluate the proposed algorithm, the authors divide 10 times the dataset set into training and testing data (10-fold cross-validation), and testing data are not available while selecting the instances. This is also done with 50% hold-out in the experiments.  $k - NN$  with different values of  $k$  and *Multilayer Perceptron* (Du and Swamy, 2014) are compared in the final evaluation. Threshold-based CNN ensemble (TE-C), threshold-based ENN ensemble (TE-E), discretization-based CNN ensemble (DE-C) and discretization-based ENN ensemble (DE-E) methods (Triguero et al., 2012) are compared with the proposed method in 31 datasets of the KEEL repository. In Kordos et al. (2019) this approach is extended to multi-output regression with the following main characteristics: multi-parent multi-point crossover with optimized numbers of splits and parents; use of up

to three populations with different initialization and mutation probabilities which can be merged into a single Pareto front, in order to reduce overfitting and improve coverage of the solution space; asymmetric mutation; efficient evaluation of the fitness function by pre-calculating and reusing the distances matrices for  $k - NN$ .

In Hamidzadeh et al. (2020), a *chaotic krill herd evolutionary algorithm* is used to optimize an unconstrained multi-objective optimization problem for instance reduction where accuracy and geometric mean are maximized in binary classification problems with imbalanced data.

The authors integrate in Rathee et al. (2019) the concept of non-dominated sorting in an adaptive search algorithm CHC (Eshelman, 1991) through an MOEA (MOCHC) for instance selection with classifiers  $k - NN$ .

In Cheng et al. (2020) an MOEA is proposed for instance selection over SVM where the objective space is divided into subregions. This evolutionary strategy based on subregions not only makes use of the individuals in each subregion for local search, but also maintains the global search capacity of the entire population. The experiments are conducted on 21 public datasets.

Finally, in Cheng et al. (2021) an MOEA for large-scale instance selection has been proposed very recently. A strategy is proposed to recursively shorten the length of each individual in the population. Two complex evolutionary operators (crossing and mutation) based on a length reduction strategy are developed to generate a population of descendants from the reduced population, together with a repair operator to repair the length of the too short individuals. The experiments are conducted on 12 public datasets.

### 3. New multi-objective constrained optimization models for instance selection

Let  $D = \{I_1, \dots, I_w\}$  be a dataset with  $w$  instances. Each instance  $I_k = (a_1^k, \dots, a_n^k, c_k)$ ,  $k = 1, \dots, w$ , has  $n$  input attributes of any type,<sup>2</sup> and one output attribute  $c_k \in \{1, \dots, s\}$ , where  $s$  is the number of output classes. We assume that at least one instance exists for each output class. Below we describe the proposed three-goal constrained boolean optimization problems for modeling wrapper and filter instance selection methods in classification tasks. For the sake of illustration, Fig. B.4 shows the complete classification process based on the proposed instance selection methods.

#### 3.1. Wrapper instance selection approach

The following three-objective constrained boolean optimization problem is proposed:

$$\begin{aligned} & \text{Maximize} && F_D^\Phi(\mathbf{x}) \\ & \text{Maximize} && F_{CV}^\Phi(\mathbf{x}, p) \\ & \text{Minimize} && C(\mathbf{x}) \\ & \text{subject to} && C(\mathbf{x}) \leq M \\ & && C(\mathbf{x}) \geq p \\ & && Q_D(\mathbf{x}) = Q_D(\mathbf{x}_D) \end{aligned} \quad (2)$$

where  $\mathbf{x} = \{x_1, x_2, \dots, x_w\}$  is a set of boolean decision variables, i.e.  $x_k \in \{true, false\}$ ,  $k = 1, \dots, w$ , being  $w$  the number of instances of the database.  $F_D^\Phi(\mathbf{x})$  function is a performance measure of a learning algorithm,<sup>3</sup>  $\Phi$  trained with the selected instances  $x_k = true$ ,  $k = 1, \dots, w$ , and tested with the full dataset  $D$ .  $F_{CV}^\Phi(\mathbf{x}, p)$  function measures the performance of a learning algorithm  $\Phi$  trained with the selected instances  $x_k = true$ ,  $k = 1, \dots, w$ , and evaluated with  $p$ -fold cross-validation<sup>4</sup> with  $2 \leq p \leq w$ . The  $C(\mathbf{x})$  function measures the number

<sup>2</sup> The data type of the input attributes is restricted by the learning algorithm used by the optimization model.

<sup>3</sup> In this paper we use the  $\Phi$  symbol to refer, interchangeably, to a learning algorithm or a classifier, since the classifier is the result of the learning algorithm.

<sup>4</sup>  $p$ -fold cross-validation is repeated (5 times as maximum) if standard deviation of mean exceeds a threshold value, by default 0.01.

of selected instances, i.e.

$$C(\mathbf{x}) = \sum_{k=1}^w \mathcal{N}(x_k)$$

where  $\mathcal{N}$  is a function that transforms a boolean value into numeric ( $true = 1$  and  $false = 0$ ). The  $M$  value is the maximum number of instances to be selected, with  $\max\{s, p\} \leq M \leq w$ . Constraint  $C(\mathbf{x}) \geq p$  ensures that instance set represented by  $\mathbf{x}$  can be evaluated with  $p$ -fold cross-validation. Constraint  $Q_D(\mathbf{x}) = Q_D(\mathbf{x}_D)$  represents that the number  $Q_D(\mathbf{x})$  of different classes in the set of selected instances  $\mathbf{x}$  is equal to the number  $Q_D(\mathbf{x}_D)$  of different classes in the full dataset  $D$ . Vector  $\mathbf{x}_D = \{1, 1, \dots, 1\}$  represents the solution with all instances of  $D$ . This constraint is only applied for classification problems. The problem (2) is therefore a *multi-objective constrained boolean optimization problem* where  $x_k = 1$  represents that instance  $x_k$  is selected, and  $x_k = 0$  represents that instance  $x_k$  is not selected, for all  $k = 1, \dots, w$ . This typically involves an *NP-hard* optimization problem where exists  $2^w$  candidate subsets of instances (including unfeasible solutions).

It is important to note that both maximization objectives are in conflict with the minimization objective in the optimization problem (2), since the fewer instances are selected, the lower (or equal) the accuracy of the classifier will be when it is evaluated in both  $D$  and  $p$ -fold cross-validation. Therefore there exists a trade-off among objectives, i.e., an improvement gained for the instance set cardinality objective (function  $C(\mathbf{x})$ ) is only achieved by worsening the accuracy objectives (functions  $F_D^{\Phi}(\mathbf{x})$  and  $F_{CV}^{\Phi}(\mathbf{x}, p)$ ).

### 3.2. Filter instance selection approach

In large datasets, wrapper instance selection methods may be impracticable due to the excessive run time required. In such situations, filter instance selection methods are an alternative to wrapper instance selection methods since the run time is considerably shorter, although the accuracy of a classifier with the selected instances may be degraded. We propose a filter instance selection method for classification tasks based on the following three-objective constrained boolean optimization problem:

$$\begin{aligned} &\text{Maximize} && C\mathcal{N}_D(\mathbf{x}) \\ &\text{Maximize} && C\mathcal{R}_D(\mathbf{x}) \\ &\text{Maximize} && C(\mathbf{x}) \\ &\text{subject to} && C(\mathbf{x}) \leq M \\ &&& Q_D(\mathbf{x}) = Q_D(\mathbf{x}_D) \end{aligned} \quad (3)$$

The function  $C\mathcal{N}_D(\mathbf{x})$  is a *consistency* measure. This measure is used for feature selection but it can be used analogously for instance selection since, by modifying the instance set, the consistency of each attribute is also altered. The intuition behind this attribute subset evaluation method is to find attributes that divide the dataset into parts with a highly predominant class. *Weka* implements the metric introduced by Liu and Setiono called *ConsistencySubsetEval* (Liu et al., 1996). This measure has been explained by H. Almuallim et al. in 1991 (Almuallim and Dietterich, 1991) and by H. Liu and R. Setiono in 1996 (Liu et al., 1996), but the research of Dash and H. Liu (the same author as before) (Dash and Liu, 2003) gives a more complete perspective. The consistency measure approach consists of finding out the best discrimination set from the original data rather than applying an algorithm to split the classes. According to *inconsistency* measure, a group of features is inconsistent when two or more instances have the same values but different labels. For example, the instances  $(1, 2, 2, a)$  and  $(1, 2, 2, b)$ , where  $a$  and  $b$  represent the class, are inconsistent. According to Dash, the consistency measure is defined by the *inconsistency rate*. The *inconsistency rate*,  $IR(S)$ , of a feature subset  $S$  is calculated as the sum of all the inconsistency counts for all the patterns divided by the total number of instances. The inconsistency count for a given pattern (the values of the selected features without the class) is

calculated as the total number of same patterns in the dataset minus the number of instances of the majority class of the pattern. For example, for the pattern  $(1, 2, 2)$  if there are 36 elements of class  $a$ , 6 for class  $b$ , and 5 for class  $c$ , the inconsistency count would be  $(36 + 6 + 5) - 36 = 11$ . Obviously, the less inconsistent the subset is, the greater the merit (consistency) of an attribute subset. The consistency of any subset can never be lower than that of the full set of attributes.

The function  $C\mathcal{R}_D(\mathbf{x})$  is a *correlation* measure used by the *Correlation-based Feature Selection* (CFS) algorithm for feature selection, which has been adapted for instance selection in this paper. CFS algorithm was developed by Mark A. Hall from the University of Waikato in Hamilton (New Zealand) throughout his doctoral thesis (Hall, 1999). This same university is well-known in the world of data science for developing the free software application *Weka* (*Waikato Environment for Knowledge Analysis*), in which Mark A. Hall has an important role as Honorary Research Associate. CFS algorithm has the advantage over *ReliefF* (Kononenko, 1994) that it generates more accurate models and reduces the number of attributes selected by half in most cases. This concept is explained years later in a summary paper (Hall, 2000). The CFS method evaluates sets of attributes instead of doing so individually. To determine the goodness of each set, the CFS algorithm evaluates how well each attribute is able to predict the class as well as the similarity degree between the attributes. In such a way that the feature sets correlated with the class and with features poorly correlated with each other obtain the higher scores. In this paper, we adapt the CFS algorithm to measure instance subsets instead of attribute subsets by using the function  $C\mathcal{R}_D(\mathbf{x})$  as follows:

$$C\mathcal{R}_D(\mathbf{x}) = \frac{n \cdot \bar{r}_{cf}(\mathbf{x})}{\sqrt{n + n \cdot (n - 1) \cdot \bar{r}_{ff}(\mathbf{x})}}$$

where  $n$  is the number of attributes of the dataset  $D$ ,  $\bar{r}_{cf}(\mathbf{x})$  is the average of the correlations between each attribute and the class attribute taking into account only the instances of  $D$  selected in  $\mathbf{x}$ , and  $\bar{r}_{ff}(\mathbf{x})$  is the average correlation between each of the  $\binom{n}{2}$  possible attribute pairs. In other words, the numerator indicates the predictive degree of a set of attributes in the selected instances, while the denominator indicates the redundancy between the attributes in the selected instances. As in CFS method, the function  $C\mathcal{R}_D(\mathbf{x})$  requires discretizing the values (usually Fayyad and Irani method).  $C\mathcal{R}_D(\mathbf{x})$  also applies the symmetrical uncertainty method to measure the degree of similarity for discrete values.

Note that, the smaller the number of instances, the greater the consistency of the dataset. Therefore the number of instances cannot be minimized. Otherwise, the population will tend to individuals with only one instance. Instead of minimizing the number of instances as in the wrapper instance selection methods, optimization problem (3) maximize the number of instances. The same happens with the redundancy of the set of instances. Note that the  $C\mathcal{R}_D(\mathbf{x})$  function measures both the correlation of the input data with the class (through the numerator of the function) and the redundancy of the input data (through the denominator). The smaller the number of selected instances, the lower the redundancy and therefore the higher the value of the COR function. Maximizing the number of instances and maximizing the  $C\mathcal{R}_D(\mathbf{x})$  function are therefore conflicting objectives. Therefore, also in the optimization problem (3) there exists a trade-off among objectives. In this way, the Pareto front will identify the sets of instances with the highest number of consistent instances, in which there is also high correlation and low redundancy.

### 3.3. Decision making for choosing the final non-dominated solution

A MOEA returns a set of non-dominated solutions that represents an approximation to the true Pareto optimal set. In order to choose a single solution, we propose an approach based on *compromise programming* (Zeleny, 1973) and *goal programming* (Jones and Tamiz, 2010; Jiménez et al., 2002). Let  $S$  be the set of non-dominated solutions of

any problem (2) or (3). We propose a decision making process to choose the non-dominated solution  $\mathbf{x}^* = \{x_1^*, x_2^*, \dots, x_w^*\} \in S$ ,  $C(\mathbf{x}^*) < w$ , such that, for all  $\mathbf{x} \in S$ :

$$\mathcal{R}_D^\Phi(\mathbf{x}^*, p) < \mathcal{R}_D^\Phi(\mathbf{x}, p)$$

∨

$$\mathcal{R}_D^\Phi(\mathbf{x}^*, p) = \mathcal{R}_D^\Phi(\mathbf{x}, p) \wedge C(\mathbf{x}^*) < C(\mathbf{x})$$

∨

$$\mathcal{R}_D^\Phi(\mathbf{x}^*, p) = \mathcal{R}_D^\Phi(\mathbf{x}, p) \wedge C(\mathbf{x}^*) = C(\mathbf{x}) \wedge C\mathcal{ON}_D(\mathbf{x}^*) > C\mathcal{ON}_D(\mathbf{x})$$

In the proposed decision making process,  $\mathcal{R}_D^\Phi(\mathbf{x}, p)$  is a piecewise function as defined in Eq. (4) given in Box I.

The interpretation of the  $\mathcal{R}_D^\Phi(\mathbf{x}, p)$  function is as follows: when a solution  $\mathbf{x}$  has not reached the  $\mathcal{F}_D^\Phi(\mathbf{x}_D)$  and  $\mathcal{F}_{CV}^\Phi(\mathbf{x}_D, p)$  goals, the  $\mathcal{R}_D^\Phi(\mathbf{x}, p)$  function measures the maximum deviation from these; on the contrary, when the goals have already been reached, the  $\mathcal{R}_D^\Phi(\mathbf{x}, p)$  function measures the average of the deviations. In this way, the *Min-Max* criterion is applied except when both goals have already been reached, in which case the arithmetic mean is used. Solutions with a number of instances less than  $p$  are evaluated with the worst possible value (value 1) since they do not allow a  $p$ -fold cross-validation. The latter will never occur in wrapper models since it is a constraint of the problem, but it can occur in filter models in which there is no such constraint.

#### 4. Multi-objective evolutionary algorithms for instance selection

In this section we describe the common components that have been implemented in the *NSGA-II* and *MODE* algorithms. These components are representation of the solutions, initialization of the population, constraint handling, evaluation of individuals and self-adaptive crossover and mutation operators. Some of these components differ slightly between the two algorithms *NSGA-II* and *MODE*. The rest of the components, such as selection, sampling and generational replacement mechanisms, are specific to each of the algorithms.

##### 4.1. Representation of solutions

*NSGA-II* uses a fixed-length representation, where each individual consists of two component blocks: the first block is a bit set for instance selection; the second block is related to self-adaptive crossover and mutation. Therefore, an individual  $I$  is represented as:

$$I = \{b_1^I, \dots, b_w^I, d_I, e_I, m_I\}$$

where  $b_i^I \in \{0, 1\}$  for  $i = 1, \dots, w$ ,  $d_I \in \{0, \dots, \delta\}$ ,  $e_I \in \{0, \dots, \epsilon\}$ , and  $m_I \in [0, 1]$ . Each bit  $b_i^I \in \{0, 1\}$  represents an instance in the dataset (1 for selected, and 0 for non-selected instances). Additionally, to carry out self-adaptive crossover and mutation, each individual has two discrete parameters  $d_I \in \{0, \dots, \delta\}$  and  $e_I \in \{0, \dots, \epsilon\}$  associated to the type of crossover and mutation, where  $\delta \geq 0$  is the number of crossover operators and  $\epsilon \geq 0$  is the number of mutation operators. Note that  $d_I = 0$  or  $e_I = 0$  means that crossover or mutation operator respectively has not been applied. Finally, the parameter  $m_I \in [0, 1]$  represents the probability of mutation for each bit  $b_i^I$ ,  $i = 1, \dots, w$ . Note that  $e_I$  indicates whether the individual  $I$  is going to be mutated or not, while  $m_I$  indicates the mutation probability of each bit in the individual  $I$ . Sections 4.2 and 4.5 indicate how these parameters are initialized and updated respectively.

In the case of *MODE*,  $d_I$  and  $e_I$  are not represented since the crossover and mutation operators are always applied. Therefore, *MODE*

only uses a real parameter  $c_I \in [0, 1]$  to represent the crossover probability. So, an individual  $I$  is represented in *MODE* as:

$$I = \{b_1^I, \dots, b_w^I, c_I\}$$

Note that in *MODE*, the crossover operator is applied independently for each bit of the mutant and parent chromosomes with the probability  $c_I$ , while in *NSGA-II* the crossover operator is uniformly applied with probability 0.5, as described in Section 4.5.

##### 4.2. Initial population

The initial population is randomly generated as described in Algorithm 6. For each individual  $I$  in the population, a number  $q \in \{1, \dots, w\}$  is first randomly generated. Next,  $q$  random bits in the individual  $I$  are fixed to 1 and the remaining  $w - q$  bits are fixed to 0. Finally,  $d_I$ ,  $e_I$  and  $m_I$  values for self-adaptive variation are randomly generated from  $\{0, \dots, \delta\}$ ,  $\{0, \dots, \epsilon\}$  and  $[0, 1]$  respectively in *NSGA-II* algorithm, while in *MODE* the parameter  $c_I$  is randomly initialized from  $[0, 1]$ . Individual  $I$  is repaired if constraint handling is based on repair method, as described in the next section.

##### 4.3. Constraint handling

We propose two different mechanisms for constraint handling. These two methods have been compared in the experiments section. In the first method, the individuals of the population are always feasible. A *repair algorithm* (Michalewicz, 1996) is applied after initialization and variation operators. The algorithm 7 shows the steps taken to repair an infeasible individual. The second method maintains feasible and infeasible individuals in the population (Jiménez et al., 1999). Infeasible individuals are ranked in positions worse than the feasible individuals, and ordered according to the *Min - Max* criterion, that is, the best infeasible individual is the one with the lowest maximum violation.<sup>5</sup> Therefore, infeasible individuals evolve toward feasibility guided by the following fitness function  $g(\mathbf{x})$ , which is minimized:

$$g(\mathbf{x}) = \begin{cases} \max\{C(\mathbf{x}) - M, p - C(\mathbf{x}), \mathcal{Q}_D(\mathbf{x}_D) - \mathcal{Q}_D(\mathbf{x})\} & \text{in problem (2)} \\ \max\{C(\mathbf{x}) - M, \mathcal{Q}_D(\mathbf{x}_D) - \mathcal{Q}_D(\mathbf{x})\} & \text{in problem (3)} \end{cases} \quad (5)$$

##### 4.4. Evaluation of individuals

The evaluation of individuals depends on the constraint handling method. With the repair-based method, all the individuals in the population are feasible after the repair, and therefore all individuals are evaluated using the three objective functions of the problem (2) or (3). When the constraint handling method is not based on repair of individuals, then infeasible individuals are evaluated using the function  $g(\mathbf{x})$ , while feasible individuals are evaluated using the three objective functions of the problem (2) or (3). Algorithms 8 and 9 show the evaluation of individuals for both constraint handling methods.

##### 4.5. Self-adaptive variation operators

*Uniform crossover* (Algorithm 12) and *bit-flip mutation* (Algorithm 14) are used as variation operators in *NSGA-II*. Therefore,  $\delta = \epsilon = 1$ . The selection of the operators<sup>6</sup> is made by means of an adaptive technique that uses the parameters  $d_I$  and  $e_I$  (in our case, both in the set  $\{0, 1\}$ ) to indicate which crossover (Algorithm 11) and which

<sup>5</sup> Note that constraints in problems (2) and (3) should be transformed into the standard formulation as in (1).

<sup>6</sup> Although the proposed algorithms for instance selection apply a single crossover operator and a single mutation operator, the proposed self-adaptive technique allows the application of any number  $\delta$  of crossover operators and any number  $\epsilon$  of mutation operators.

$$\mathcal{R}_D^\Phi(\mathbf{x}, p) = \begin{cases} 1 & \text{if } C(\mathbf{x}) < p \\ \frac{\mathcal{F}_D^\Phi(\mathbf{x}_D) - \mathcal{F}_D^\Phi(\mathbf{x}) + \mathcal{F}_{CV}^\Phi(\mathbf{x}_D, p) - \mathcal{F}_{CV}^\Phi(\mathbf{x}, p)}{2} & \text{if } (\mathcal{F}_D^\Phi(\mathbf{x}_D) - \mathcal{F}_D^\Phi(\mathbf{x})) \leq 0 \wedge (\mathcal{F}_{CV}^\Phi(\mathbf{x}_D, p) - \mathcal{F}_{CV}^\Phi(\mathbf{x}, p)) \leq 0 \\ \max \{ \mathcal{F}_D^\Phi(\mathbf{x}_D) - \mathcal{F}_D^\Phi(\mathbf{x}), \mathcal{F}_{CV}^\Phi(\mathbf{x}_D, p) - \mathcal{F}_{CV}^\Phi(\mathbf{x}, p) \} & \text{otherwise} \end{cases} \quad (4)$$

Box I.

mutation (Algorithm 13) is carried out on the individual  $I$ . Since there is only one crossover operator and only one mutation operator,  $d_I = 0$  or  $e_I = 0$  represents that the crossover or mutation operator respectively will not be applied, and  $d_I = 1$  or  $e_I = 1$  represent that the uniform crossover operator or bit-flip mutation operator respectively will be applied (see Algorithms 11 and 13). The algorithm 10 is used to generate two children from two parents by self-adaptive crossover and mutation. A probability of variation  $p_v = 0.1$  is first set, and, then, for each individual in each generation, a Bernoulli random variable with probability  $p_v$  is asked whether or not the values  $d_I$  and  $e_I$  have to be changed. In the particular case of crossover, fixed the first selected individual  $I$ , the decision on whether the crossover takes place or not depends on the value of  $d_I$ . In summary, if an individual comes from a given crossover or a given mutation, that specific crossover and mutation is preserved to their offspring unless the Bernoulli random variable returns *true*. For this reason,  $p_v$  must be a small value to ensure a controlled evolution. Although the probability of the crossover and mutation is not explicitly represented, it can be computed as the ratio the individuals for which crossover and mutation values are set to 1. In the case of bit-flip mutation, a parameter  $m_I$  is additionally used to indicate the probability of mutation for each bit of the individual  $I$ . Again, a Bernoulli random variable with probability  $p_v$  is asked whether or not the value  $m_I$  has to be changed.

In the case of *MODE*, we use the variant *DE/best/1/bin*. So, the base individual is the best individual of the population,<sup>7</sup> and one pair of random individuals is chosen for *binary differential mutation* (Wagdy, 2016), along with *binomial crossover*. Algorithm 15 describes the full variation process. Mutation, which is always applied in *MODE*, is showed in Algorithm 16. Crossover is also always applied between mutant and parent, bit-to-bit with the self-adaptive probability  $c_I$  of the parent (Algorithm 17).

## 5. Experiments and results

This section shows the experiments carried out and the results. The datasets used in the experiments are described, a nomenclature is established to name each method to facilitate their identification, and the results of three blocks of experiments are shown. In the first block, the performance of the MOEAs is tested using the *hypervolume metric*. In the second block, the performance of the classification models obtained with the MOEAs for instance selection are tested using *accuracy* and *retention percentage* performance metrics on *10-fold cross-validation*, and the results are compared with the results obtained with 12 instance selection methods from the *KEEL* software (Alcalá-Fdez et al., 2010). Finally, in the third block of experiments, the solution of each MOEA on the 100% of the data is evaluated and compared with 12 instance selection methods from the *KEEL* software using the proposed decision making process. Fig. B.5 summarizes the main characteristics of these three blocks of experiments. It is important to mention that we have not been able to compare the techniques proposed in this paper with those described in Section 2.3 because the source codes are not available. However, the set of instance selection methods available in *KEEL* is

Table 1

Datasets for classification used in this paper.

Dataset	Number of instances	Number of attributes	Attribute type	Number of classes
WINE-QUALITY-RED	1599	12	Real	6
WINE-QUALITY-WHITE	4898	12	Real	7
ONLINE-SHOPPERS	12330	18	Real, Integer	2

quite representative, and many of the methods described in Section 2.3 also use the instance selection methods available in *KEEL* for their comparisons.

### 5.1. Datasets

To carry out the experiments, three public datasets with an increasing number of instances have been considered. We have preferred to use public datasets for reasons of reproducibility of results. Likewise, we have preferred to use a reduced set of datasets in order to carry out a more detailed analysis of them. *Wine quality — red*, *wine quality — white* and *online shoppers purchasing intention* datasets from the *UCI Machine Learning Repository* (Dua and Graff, 2017) have been used in this paper. The first two are datasets for multi-class classification with real attributes while the third is a dataset for binary classification with real and integer attributes. Table 1 summarizes the characteristics of the three datasets.

### 5.2. Nomenclature

In order to facilitate the identification of the multi-objective evolutionary methods for which the results will be shown, we use the following pattern:

MOEA - Method type - Number of objectives - Evaluator - Constraint handling method - Minimum retention percentage

where:

- MOEA is *NSGA-II* or *MODE*.
- Method type is *W* for wrapper instance selection method and *F* for filter instance selection method.
- Number of objectives is *3o* for 3-objective optimization models and *2o* for 2-objective optimization models.
- Evaluator is the classification algorithm followed of a performance metric, for wrapper method, or the measures *CON-COR* for filter method.
- Constraint handling method is *R* for repair method and *MM* for *Min – Max* criterion based method.
- Minimum reduction percentage is a number that represents the minimum reduction percentage of instances imposed as constraint.

For example, the name *MODE-W-3o-C4.5-ACC-R-0* refers to *MODE* with 3 objectives that implements an wrapper instance selection method that uses the *C4.5* classifier evaluated with the performance metric of

<sup>7</sup> We choose a random non-dominated individual.



**Table 2**

Evolutionary methods used for each dataset in the hypervolume test.

Method	Datasets
NSGA-II-W-3o-INN-ACC-R-0	WINE-QUALITY-RED
MODE-W-3o-INN-ACC-R-0	WINE-QUALITY-RED
NSGA-II-W-3o-INN-ACC-MM-0	WINE-QUALITY-RED
MODE-W-3o-INN-ACC-MM-0	WINE-QUALITY-RED
NSGA-II-W-3o-INN-ACC-R-50	WINE-QUALITY-RED
MODE-W-3o-INN-ACC-R-50	WINE-QUALITY-RED
NSGA-II-W-3o-INN-ACC-MM-50	WINE-QUALITY-RED
MODE-W-3o-INN-ACC-MM-50	WINE-QUALITY-RED
NSGA-II-F-3o-CON-COR-R-0	WINE-QUALITY-WHITE, ONLINE-SHOPPERS
MODE-F-3o-CON-COR-R-0	WINE-QUALITY-WHITE, ONLINE-SHOPPERS
NSGA-II-F-3o-CON-COR-MM-0	WINE-QUALITY-WHITE, ONLINE-SHOPPERS
MODE-F-3o-CON-COR-MM-0	WINE-QUALITY-WHITE, ONLINE-SHOPPERS
NSGA-II-F-3o-CON-COR-R-50	WINE-QUALITY-WHITE, ONLINE-SHOPPERS
MODE-F-3o-CON-COR-R-50	WINE-QUALITY-WHITE, ONLINE-SHOPPERS
NSGA-II-F-3o-CON-COR-MM-50	WINE-QUALITY-WHITE, ONLINE-SHOPPERS
MODE-F-3o-CON-COR-MM-50	WINE-QUALITY-WHITE, ONLINE-SHOPPERS

accuracy (optimization model (2)), with repair method for constraint handling, and minimum reduction percentage of 0%. The name *NSGA-II-F-3o-CON-COR-MM-50* refers to *NSGA-II* with 3 objectives for the filter instance selection method based on consistency and correlation (optimization model (3)), with constraint handling based on *Min-Max* criterion, and minimum reduction percentage of 50%. Figs. B.6 and B.7 compiles the instance selection methods proposed in this document and their names.

### 5.3. Hypervolume test

The goal of this block of experiments is to compare the performance of the *NSGA-II* and *MODE* algorithms without limitation of the reduction of the set of instances and with limitation, as well as to compare the proposed techniques for constraint handling. To do this, *NSGA-II* and *MODE* have been run 10 times with the configurations of problems (2) and (3) that appear in Table 2. Due to the excessive run time required by the wrapper instance selection methods with large datasets, they have been executed only with the *WINE-QUALITY-RED* dataset with a number of folds  $p = 5$  for cross-validation in the second objective function. Filter instance selection methods have been applied to the datasets *WINE-QUALITY-WHITE* and *ONLINE-SHOPPERS*.

We have used the *hypervolume metric* (Zitzler et al., 2002) to measure the performance of the algorithms. This metric measures both the optimality and the diversity of the set of non-dominated solutions. The hypervolume is defined as the volume of the search space dominated by a population  $P$ , formulated as:

$$HV(P) = \bigcup_{i=1}^{|Q|} v_i \quad (6)$$

where  $Q \subseteq P$  is the set of non-dominated individuals of  $P$ , and  $v_i$  is the volume of the individual  $i$ . Subsequently, the *hypervolume ratio* (HVR) is defined as the ratio of the volume of the non-dominated search space over the volume of the entire search space, formulated as:

$$HVR(P) = 1 - \frac{HV(P)}{VS} \quad (7)$$

where  $VS$  is the volume of the search space. Computing HVR requires reference points that identify the maximum and minimum values for each objective function. For optimization problem (2), the following reference points  $(F_{min}^{\Phi,D}, F_{min}^{\Phi,CV}, C_{min})$  and  $(F_{max}^{\Phi,D}, F_{max}^{\Phi,CV}, C_{max})$  are set:

$$\begin{aligned} F_{min}^{\Phi,D} &= 0, & F_{max}^{\Phi,D} &= 1, & F_{min}^{\Phi,CV} &= 0, \\ F_{max}^{\Phi,CV} &= 1, & C_{min} &= 1, & C_{max} &= w \end{aligned}$$

In the case of the optimization problem (3), the reference points  $(C\mathcal{O}\mathcal{N}_{min}^D, C\mathcal{O}\mathcal{R}_{min}^D, C_{min})$  and  $(C\mathcal{O}\mathcal{N}_{max}^D, C\mathcal{O}\mathcal{R}_{max}^D, C_{max})$  are setting as

follows:

$$\begin{aligned} C\mathcal{O}\mathcal{N}_{min}^D &= 0, & C\mathcal{O}\mathcal{N}_{max}^D &= 1, & C\mathcal{O}\mathcal{R}_{min}^D &= 0, & C\mathcal{O}\mathcal{R}_{max}^D &= 1, \\ C_{min} &= 1, & C_{max} &= w \end{aligned}$$

Figs. D.8–D.10 graphically show the confidence intervals of the hypervolume over 10 runs of the proposed evolutionary methods with the datasets *WINE-QUALITY-RED*, *WINE-QUALITY-WHITE* and *ONLINE-SHOPPERS* respectively. Confidence intervals are closely related to statistical significance testing (Cumming and Calin-Jageman, 2017), and overlapping confidence intervals indicate that there are no statistically significant differences between the algorithms.

### 5.4. Performance test of classification models

In this second block of experiments, the performance of the classifiers built with the datasets reduced by the proposed evolutionary methods is compared, together with 12 non-evolutionary methods executed with the *KEEL* software. In addition to the methods used in the hypervolume test shown in Table 2, the two-objective methods *NSGA-II-W-2o-INN-ACC-R-0*, *MODE-W-2o-INN-ACC-R-0*, *NSGA-II-W-2o-INN-ACC-MM-0* and *MODE-W-2o-INN-ACC-MM-0* have been included in these experiments to compare the performance of the classifiers obtained with the three-objective and two-objective wrapper methods. For two-objective methods, the maximization of the objective function  $F_{CV}^{\Phi}(x, p)$  has been removed. Three-objective filter instance selection methods have also been included for the case of the *WINE-QUALITY-RED* dataset. Regarding non-evolutionary methods, we have included in the experiments the 12 methods implemented in the *Training Set Selection* (TSS) category of the *KEEL* software, shown in Table 3.

The evolutionary methods proposed together with the non-evolutionary methods have been evaluated using stratified 10-fold cross-validation. In this way, the classifiers trained on the instance sets selected by each instance selection method (named  $T$ ) are tested on instance sets unseen by the instance selection method. As a result of the stratified 10-fold cross-validation process illustrated graphically in Fig. 1, the following 5 measurements are obtained:

1.  $\overline{ACC}_D^{1-NN}$ : Average accuracy, over 10 runs, of the  $1 - NN$  classifiers trained in the instance set  $T$  selected by the instance selection method in the corresponding 9 folds and evaluated in the instance set  $D$  formed by the 9 corresponding folds.
2.  $\overline{ACC}_{CV}^{1-NN}$ : Average accuracy, over 10 runs, of the  $1 - NN$  classifiers evaluated, in turn, in  $p$ -fold cross-validation<sup>8</sup> over the instance set  $T$  selected by the instance selection method in the corresponding 9 folds.
3.  $\overline{ACC}_E^{1-NN}$ : Average accuracy, over 10 runs, of the  $1 - NN$  classifiers trained in the instance set  $T$  selected by the instance selection method in the corresponding 9 folds and evaluated in the corresponding unseen test fold  $E$ .
4.  $RET$ : Average percentage, over 10 runs, of the number of instances selected by the instance selection method in the 9 corresponding folds.

Tables C.19–C.21 show the results obtained through the 10-fold cross-validation process. In these tables, the separator lines distinguish three groups of methods: (1) proposed evolutionary instance selection methods without imposing minimum reduction of instances (top); (2) proposed evolutionary instance selection methods imposing a minimum instance reduction of 50% (middle); and (3) instance selection methods implemented in the *KEEL* software (bottom). Statistical tests have been performed to detect statistically significant differences between the compared methods in each of the metrics  $\overline{ACC}_D^{1-NN}$ ,  $\overline{ACC}_{CV}^{1-NN}$ ,  $\overline{ACC}_E^{1-NN}$  and  $RET$  in each of the 10 folds.<sup>9</sup> First, the Shapiro–Wilks

<sup>8</sup> We have used  $p = 5$  as in the optimization phase.

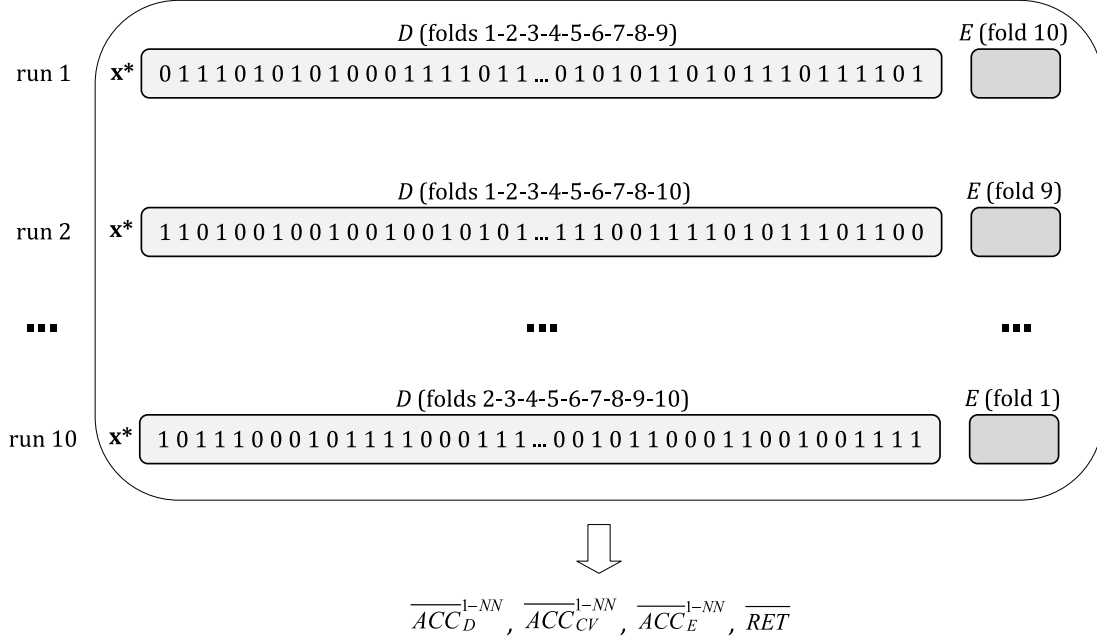
<sup>9</sup> We have not performed statistical tests on the metric  $\overline{ACC}_T^{1-NN}$  because all the compared methods have obtained an accuracy of 1.00 in all the folds.



**Table 3**

Non-evolutionary instance selection methods used in this paper.

Short Name	Full Name	Reference
<i>AllKNN-TSS</i>	All $k$ -Nearest Neighbors	(Tomek, 1976)
<i>ENN-TSS</i>	Edited Nearest Neighbor	(Wilson, 1972)
<i>ENNT<sub>h</sub>-TSS</i>	Edited Nearest Neighbor with Estimation of Probabilities Threshold	(Vazquez et al., 2005)
<i>ENRBF-TSS</i>	Edited Normalized Radial Basis Function	(Grochowski and Jankowski, 2004a)
<i>MENN-TSS</i>	Modified Edited Nearest Neighbor	(Hattori and Takahashi, 2000)
<i>ModelCS-TSS</i>	Model Class Selection	(Brodley, 1993)
<i>MultEdit-TSS</i>	Multiple Edited Nearest Neighbor	(Devijver, 1986)
<i>NCNEdit-TSS</i>	Nearest Centroid Neighborhood Edition	(Sánchez et al., 2003)
<i>POP-TSS</i>	Pattern by Ordered Projections	(Riquelme et al., 2003b)
<i>PSRCG-TSS</i>	Prototype Selection by Relative Certainty Gain	(Sebban et al., 2002)
<i>RNG-TSS</i>	Prototype Selection based on Relative Neighborhood Graphs	(Sánchez et al., 1997)
<i>VSM-TSS</i>	Variable Similarity Metric	(Lowe, 1995)

**Fig. 1.** Stratified 10-fold cross-validation for instance selection.**Table 4**Performance of the 1-NN, C4.5 and Random Forest classifiers using the complete dataset  $D$  (solution  $\mathbf{x}_D$ ).

Dataset	$ACC_D^{1-NN}(\mathbf{x}_D)$	$ACC_{CV}^{1-NN}(\mathbf{x}_D, 5)$	$ACC_D^{C4.5}(\mathbf{x}_D)$	$ACC_{CV}^{C4.5}(\mathbf{x}_D, 5)$	$ACC_D^{RF}(\mathbf{x}_D)$	$ACC_{CV}^{RF}(\mathbf{x}_D, 5)$
WINE-QUALITY-RED	1.0000	0.6373	0.9099	0.6060	1.0000	0.7079
WINE-QUALITY-WHITE	1.0000	0.6384	0.8971	0.5874	1.0000	0.6872
ONLINE-SHOPPERS	1.0000	0.8144	0.9419	0.8936	1.0000	0.9004

test has been performed to test the null hypothesis stating that the samples of each method came from a normally distributed population. Using a 95% confidence level, some samples failed to accept the null hypothesis. Thus, to detect if statistically significant differences between models exist, the non-parametric pairwise Wilcoxon signed-rank, with Bonferroni correction, has been applied. Then, we compute the times each method has won (wins), the times that each method has lost (losses) and the difference (wins – losses). Tables C.22–C.24 show the win–loss rankings of these statistical tests for each of the four metrics in the datasets WINE-QUALITY-RED, WINE-QUALITY-WHITE and ONLINE-SHOPPERS respectively.

### 5.5. Solution report

Finally, in this third block of experiments we report the evaluation of the classification model obtained by each of the proposed instance selection methods, together with the 12 non-evolutionary methods of the KEEL software, considering the complete dataset  $D$  in order to identify the best definitive solution for the final implementation of

the classification system. In this experiment, each instance selection method has been executed only once using the same seed in all cases. We report the evaluation of each solution  $\mathbf{x}^*$  in the following metrics:

1.  $ACC_T^\Phi(\mathbf{x}^*)$ : Accuracy of the  $\Phi$  classifier trained in the instance set  $T$  (solution  $\mathbf{x}^*$ ) selected by the instance selection method.<sup>10</sup> from the complete dataset  $D$  and evaluated in the same instance set  $T$ . This metric is not shown when  $\Phi = 1-NN$  since in this case it is always 1.0.
2.  $ACC_D^\Phi(\mathbf{x}^*)$ : Accuracy of the  $\Phi$  classifier trained in the instance set  $T$  (solution  $\mathbf{x}^*$ ) selected by the instance selection method from the complete dataset  $D$  and evaluated in the complete dataset  $D$ .

<sup>10</sup> Note that the instance set  $T$  corresponds to the instance set such that  $\mathbf{x}_k^* = true$  (and therefore  $\mathcal{N}(\mathbf{x}_k^*) = 1$ ),  $k = 1, \dots, w$ , in the non-dominated solution  $\mathbf{x}^*$  of the constrained multi-objective optimization problems (2) and (3).

3.  $ACC_{CV}^{\Phi}(\mathbf{x}^*, 5)$ : Accuracy of the  $\Phi$  classifier evaluated in 5-fold cross-validation over the instance set  $T$  (solution  $\mathbf{x}^*$ ) selected by the instance selection method from the complete dataset  $D$ .
4.  $R_D^{\Phi}(\mathbf{x}^*, 5)$ : Value of the performance metric, proposed in this paper in Eq. (4) (with the  $\Phi$  classifier and  $p = 5$ ), for the solution  $\mathbf{x}^*$  obtained by the instance selection method from the complete dataset  $D$ . Although the  $R_D^{\Phi}(\mathbf{x}^*, p)$  function has been used in this paper to choose the non-dominated solution from the final population of a MOEA, it is a general performance metric that can therefore also be used to compare instance sets obtained with any instance selection method.
5.  $\mathcal{RET}(\mathbf{x}^*)$ : Percentage of the number of instances<sup>11</sup> of the solution  $\mathbf{x}^*$  selected by the instance selection method from the complete dataset  $D$ .

In addition to the  $1 - NN$  classifier, in this block of experiments the  $C4.5$  (Quinlan, 2014) and *Random Forest* (RF) (Breiman, 2001) classifiers have been used in the wrapper instance selection methods. The  $C4.5$  and RF algorithms are appropriate for use in wrapper instance selection methods as they do not require excessive training times. Furthermore,  $C4.5$  allows to interpret the classification, and RF is characterized by the high accuracy of its classifiers. The solutions obtained with each classifier are also evaluated using the other classifiers, in order to measure the robustness of the solutions. Table 4 shows accuracy values for the *WINE-QUALITY-RED*, *WINE-QUALITY-WHITE* and *ONLINE-SHOPPERS* complete datasets with the  $1 - NN$ ,  $C4.5$  and RF classifiers. In Table 4, vector  $\mathbf{x}_D = \{1, 1, \dots, 1\}$  represents the solution with all instances of  $D$  selected, that is, the complete dataset  $D$ . The values reported in Table 4 are significant since they are the goals to be achieved by the solutions of the different instance selection methods compared using the  $R_D^{\Phi}(\mathbf{x}^*, p)$  function.

Tables C.25–C.27 show the results obtained in this block of experiments. Again, the separator lines in the tables distinguish the three groups of methods analyzed (evolutionary instance selection methods without retention limitation, evolutionary instance selection methods with 50% retention limitation, and non-evolutionary instance selection methods). Method with best value of  $R_D^{\Phi}(\mathbf{x}^*)$  in each of the classifiers used has been marked in bold.

### 5.6. Notes on fairness in comparisons

It is important to note that the comparisons made were completely fair. Although evolutionary methods have been run on the Weka platform while non-evolutionary methods have been run on the *KEEL* platform, the following actions were taken to ensure fairness in comparisons:

- For the stratified 10-fold cross-validation statistical tests in Section 5.4, 10 partitions have been generated with the *KEEL* software (in .dat format), and these have been converted to .arff format (Weka format). So our evolutionary methods (both three and two objectives) have run on exactly the same partitions as the non-evolutionary methods. Once the reduced datasets were obtained with the non-evolutionary methods, these have been transformed into arff format and have been used for their evaluations, together with the reduced datasets obtained with the evolutionary methods, using the same classification algorithm of the Weka platform ( $1 - NN$ ) and the same test partitions.
- For the comparisons considering the complete dataset  $D$  of Sections 5.5, 7.2 and 7.3, the reduced datasets obtained with the *KEEL* software have been transformed into arff format and have been trained with the same classification algorithms from Weka ( $1 - NN$ ,  $C4.5$  and *Random Forest*) and the same hyperparameters as the evolutionary methods of three and two objectives.

Table 5

*NSGA-II* vs. *MODE*: win-loss hypervolume ranking test.

MOEA	Wins	Losses	Difference
<i>NSGA-II</i>	0	2	−2
<i>MODE</i>	2	0	2

Table 6

*NSGA-II* vs. *MODE*: mean win-loss stratified 10-fold cross-validation performance test.

MOEA	$ACC_D^{1-NN}$	$ACC_{CV}^{1-NN}$	$ACC_E^{1-NN}$	$RET$
<i>NSGA-II</i>	6	−9, 3	1, 8	−1, 75
<i>MODE</i>	7, 4	−5, 15	3, 3	−4, 3

Table 7

*NSGA-II* vs. *MODE*: Number of times the algorithm obtains the best value of  $R_D^{1-NN}(\mathbf{x}^*)$ ,  $R_D^{C4.5}(\mathbf{x}^*)$ ,  $R_D^{RF}(\mathbf{x}^*)$ , or  $\mathcal{RET}(\mathbf{x}^*)$ .

MOEA	$R_D^{1-NN}(\mathbf{x}^*)$	$R_D^{C4.5}(\mathbf{x}^*)$	$R_D^{RF}(\mathbf{x}^*)$	$\mathcal{RET}(\mathbf{x}^*)$
<i>NSGA-II</i>	5	5	3	5
<i>MODE</i>	2	2	4	2

- For the hypervolume tests of Section 5.1, all evolutionary methods have been executed under exactly the same conditions (10 runs with the same seeds in the evolutionary algorithms, classification algorithm, and dataset partitioning for 5-fold cross-validation).

## 6. Analysis of results and discussion

Based on the results obtained with the experiments, in this section we will answer the following questions: Which MOEA is better to solve the proposed optimization problems for instance selection? Which constraint handling method is most efficient? Is the 3-objective optimization model better than the 2-objective optimization model for instance selection? Do evolutionary instance selection methods outperform non-evolutionary instance selection methods? The following sections analyze each of these questions separately.

### 6.1. *NSGA-II* vs. *MODE*

The hypervolume confidence intervals shown in Section 5.3 allow us to analyze the performance of the *NSGA-II* and *MODE* optimization algorithms in terms of optimality and diversity of the non-dominated solutions found by both algorithms. Additionally, with the cross-validation statistical tests shown in Section 5.4, we can compare the performance of the classification models built from the reduced databases obtained with both *NSGA-II* and *MODE* algorithms.

To analyze the results obtained with the hypervolume confidence intervals, we have carried out a ranking with the methods where each method is assigned the difference between the times it has had statistically significant differences for and against (wins and losses). Table 5 shows the ranking, over the three datasets analyzed, of the algorithms *NSGA-II* and *MODE*. As we can see in Table 5, *NSGA-II* has lost 2 times against *MODE* in a total of 12 comparisons (once with the *WINE-QUALITY-WHITE* dataset and once with the *ONLINE-SHOPPERS* dataset, the rest of the time they were tied), so we can conclude that *MODE* has a better behavior than *NSGA-II* with respect to the optimality and diversity of the solutions generated.

A second way to compare the *NSGA-II* and *MODE* algorithms is through the performance of the classifiers built with the reduced databases obtained with both algorithms. For this we will take into account the results of experiments shown in Section 5.4 (win-loss statistical tests) and in Section 5.5 (final solutions obtained from the complete dataset  $D$ ).

<sup>11</sup> Note that  $\mathcal{RET}(\mathbf{x}^*) = \frac{C(\mathbf{x}^*)}{uc} \cdot 100$ .

**Table 8**Repair method vs. *Min – Max* criterion: Win-loss hypervolume and run time tests.

Constraint handling method	Hypervolume			Run time		
	Wins	Losses	Difference	Wins	Losses	Difference
<i>Repair</i>	7	0	7	2	6	–4
<i>Min – Max</i>	0	7	–7	6	2	4

- *Win-loss statistical tests:* Table 6 shows the mean of the win-loss ranking, taken from Tables C.19–C.21, for the metric  $ACC_D^{1-NN}$ ,  $ACC_{CV}^{1-NN}$ ,  $ACC_E^{1-NN}$  and  $RET$ , in each of the *NSGA-II* and *MODE* algorithms. We can see that *MODE* outperforms *NSGA-II* in all three performance metrics  $ACC_D^{1-NN}$ ,  $ACC_{CV}^{1-NN}$  and  $ACC_E^{1-NN}$ . However, *NSGA-II* gets slightly smaller databases, which may be due to the selective pressure produced by the binary tournament selection, which is not applied in the *MODE* algorithm.
- *Final solutions obtained from the complete dataset D:* In this case we use the  $R_D^{\Phi}(x^*, p)$  and  $RET(x^*)$  metrics to compare the final solutions  $x^*$  obtained with the *NSGA-II* and *MODE* algorithms in all their variants. Table 7 shows the number of times that the *NSGA-II* and *MODE* algorithms have obtained the best value of  $R_D^{1-NN}(x^*)$ ,  $R_D^{C4.5}(x^*)$ ,  $R_D^{RF}(x^*)$  or  $RET(x^*)$  (not necessarily simultaneously) in their corresponding group of methods considering the three datasets *WINE-QUALITY-RED*, *WINE-QUALITY-WHITE* and *ONLINE-SHOPPERS*. The results show that *NSGA-II* outperforms *MODE* in this block of experiments (except for the  $R_D^{RF}(x^*)$  metric where the results are similar), unlike the results obtained with the hypervolume confidence intervals and with the win-loss statistical tests, where *MODE* outperformed *NSGA-II*. This is a sign that *NSGA-II* is more prone to overfitting than *MODE*, produced mainly by the low selective pressure of *MODE* as it lacks a selection algorithm, which allows greater diversity and a better performance in terms of statistical significance. The differences are reduced with the  $R_D^{RF}(x^*)$  metric since RF is less prone to overfitting than 1 – NN and C4.5.

### 6.2. Repair based constraint handling method vs. *Max – Min* criterion based constraint handling method

Another important issue in this research is the analysis of the two constraint management techniques proposed for instance selection. Table 8 shows the win-loss ranking for the constraint handling technique based on repair algorithm versus that based on the *Min–Max* criterion, for hypervolume and run time performance metrics. These ranking tests clearly show that the repair technique produces better hypervolume than the *Min – Max* technique, however at the expense of a longer run time.

We have also analyzed the classification models obtained from the reduced datasets with both constraint handling techniques. For this, win-loss statistical tests and final solution obtained with the complete dataset *D* are analyzed:

- *Win-loss statistical tests:* We have calculated the mean win-loss ranking of the three problems *WINE-QUALITY-RED*, *WINE-QUALITY-WHITE* and *ONLINE-SHOPPERS* (shown in Tables C.22–C.24) in each of the metrics  $ACC_D^{1-NN}$ ,  $ACC_{CV}^{1-NN}$ ,  $ACC_E^{1-NN}$  and  $RET$ . The results of the mean win-loss ranking tests, shown in Table 9, do not indicate in this case a clear difference in favor or against between both constraint handling methods, and therefore it can be concluded that the performance of the classification models obtained with both methods is similar.
- *Final solutions obtained from the complete dataset D:* Table 10 shows the number of times each constraint handling method has obtained the best value of  $R_D^{1-NN}(x^*)$ ,  $R_D^{C4.5}(x^*)$ ,  $R_D^{RF}(x^*)$  or  $RET(x^*)$  (separately) in their corresponding group of methods considering the three datasets *WINE-QUALITY-RED*, *WINE-QUALITY-WHITE* and *ONLINE-SHOPPERS*. We can see that, in

**Table 9**Repair method vs. *Min – Max* criterion: Mean win-loss stratified 10-fold cross-validation performance test.

Constraint handling method	$ACC_D^{1-NN}$	$ACC_{CV}^{1-NN}$	$ACC_E^{1-NN}$	$RET$
<i>Repair</i>	5,7	–8,45	2,4	–2,4
<i>Min – Max</i>	7,7	–6	2,7	–3,65

**Table 10**Repair method vs. *Min – Max* criterion: Number of times the algorithm obtains the best value of  $R_D^{1-NN}(x^*)$ ,  $R_D^{C4.5}(x^*)$ ,  $R_D^{RF}(x^*)$ , or  $RET(x^*)$ .

Constraint handling method	$R_D^{1-NN}(x^*)$	$R_D^{C4.5}(x^*)$	$R_D^{RF}(x^*)$	$RET(x^*)$
<i>Repair</i>	1	3	0	4
<i>Min – Max</i>	6	4	7	3

**Table 11**

3-objective optimization model vs. 2-objective optimization model (only for wrapper instance selection methods): Mean win-loss stratified 10-fold cross-validation performance test.

Optimization model	$ACC_D^{1-NN}$	$ACC_{CV}^{1-NN}$	$ACC_E^{1-NN}$	$RET$
3-objective	10,5	–6,625	5,125	–5,125
2-objective	7,125	–23,625	4,75	7,125

general, the *Min – Max* method performs better than the repair method. However, the *Min – Max* technique produced worse hypervolume than the repair method. This need not be contradictory. The explanation is as follows: the *Max – Min* method produces less diversity in feasible solutions than the repair method, and that worsens the hypervolume that measures both optimality and diversity, but the final solution chosen is more accurate with the *MaxMin* method than with the repair method.

### 6.3. 3-objective optimization model vs. 2-objective optimization model

One of the main contributions of this work is the incorporation of a third objective in the wrapper methods to maximize the performance of the classifier in *p*-fold cross-validation on the reduced set of instances. To analyze the effect of this third objective, we have compared the classification models obtained with two-objective wrapper methods and with three-objective wrapper methods.

- *Win-loss statistical tests:* Table 11 shows the mean win-loss rankings for both types of instance selection methods (two-objective and three-objective wrapper methods shown in Table C.22). As expected, the mean ranking for the metric  $ACC_{CV}^{1-NN}$  is better in three-objective methods than in two-objective methods. But in addition, due to the generalization ability added by the third objective, the three-objective methods are also better than the two-objective methods for the  $ACC_D^{1-NN}$  and  $ACC_E^{1-NN}$  metrics. However, this generalization ability also implies that the three-objective methods reduce fewer instances than the two-objective methods.
- *Final solutions obtained from the complete dataset D:* Table 12 shows the number of times each constraint handling method has obtained the best value of  $R_D^{1-NN}(x^*)$ ,  $R_D^{C4.5}(x^*)$ ,  $R_D^{RF}(x^*)$  or  $RET(x^*)$  (separately) in their corresponding group of methods considering the dataset *WINE-QUALITY-RED*. The results show the best performance of the three-objective methods in all metrics except instance retention. These results are in line with those obtained through the win-loss statistical tests.



**Table 12**

3-objective optimization model vs. 2-objective optimization model (only for wrapper instance selection methods): Number of times the algorithm obtains the best value of  $\mathcal{R}_D^{1-NN}(\mathbf{x}^*)$ ,  $\mathcal{R}_D^{C4.5}(\mathbf{x}^*)$ ,  $\mathcal{R}_D^{RF}(\mathbf{x}^*)$ , or  $\mathcal{RET}(\mathbf{x}^*)$ .

Optimization model	$\mathcal{R}_D^{1-NN}(\mathbf{x}^*)$	$\mathcal{R}_D^{C4.5}(\mathbf{x}^*)$	$\mathcal{R}_D^{RF}(\mathbf{x}^*)$	$\mathcal{RET}(\mathbf{x}^*)$
3-objective	2	2	2	0
2-objective	0	0	0	2

**Table 13**

Evolutionary instance selection methods vs. non-evolutionary instance selection methods: mean win-loss stratified 10-fold cross-validation performance test.

Method type	$ACC_D^{1-NN}$	$ACC_{CV}^{1-NN}$	$ACC_E^{1-NN}$	RET
Evolutionary with retention constraint	-5,7	-8,1	-9,75	12,7
Evolutionary without retention constraint	19,1	-6,35	14,85	-18,75
Non-evolutionary	-7,4	8,03	-2,83	3,36

#### 6.4. Evolutionary instance selection methods vs. non-evolutionary instance selection methods

Finally, in this section we compare multi-objective evolutionary approaches, with and without retention constraint, with non-evolutionary approaches.

- **Win-loss statistical tests:** Table 13 shows the mean win-loss rankings for evolutionary methods, with 50% retention constraint and without retention constraint, and for non-evolutionary methods, of the three problems WINE-QUALITY-RED, WINE-QUALITY-WHITE and ONLINE-SHOPPERS in each of the metrics  $ACC_D^{1-NN}$ ,  $ACC_{CV}^{1-NN}$ ,  $ACC_E^{1-NN}$  and RET. Looking at the main metric  $ACC_E^{1-NN}$ , which measures the performance of classifiers on unseen external data, evolutionary methods clearly outperform non-evolutionary methods when the retention constraint is not imposed. We can also observe that the evolutionary methods with the 50% retention constraint retain more instances than the non-evolutionary methods while maintaining a better performance for the metric  $ACC_D^{1-NN}$ . In general, no group of methods is better than the rest for all the metrics evaluated, and classifiers with higher accuracy usually imply a lower percentage of instance reduction. However, the evolutionary approaches proposed in this work allow the instance selection method to be configured according to the requirements of the learning environment by imposing instance retention constraints. Non-evolutionary methods reduce the set of instances more than evolutionary methods when the latter are configured without limitation on the size of the set of instances. This greater reduction in the size of the set of instances also produces, however, a greater degradation of the accuracy in the classification models obtained with the non-evolutionary methods when these are evaluated in the complete set of instances  $D$  (metric  $ACC_D^{1-NN}$ ). This indicates that non-evolutionary methods are eliminating more useful instances than non-evolutionary methods. Another consequence of the greater reduction in the instance set size that occurs in non-evolutionary methods is an increase in the accuracy of the classification models in cross-validation mode (metric  $ACC_{CV}^{1-NN}$ ). This is due to the fact that the smaller the number of selected instances, the classification algorithms produce a better generalization error since the test folds have already been seen by the instance selection algorithm.
- **Final solutions obtained from the complete dataset  $D$ :** Table 14 shows the number of times each type of method has obtained the best value of  $\mathcal{R}_D^{1-NN}(\mathbf{x}^*)$ ,  $\mathcal{R}_D^{C4.5}(\mathbf{x}^*)$ ,  $\mathcal{R}_D^{RF}(\mathbf{x}^*)$  or  $\mathcal{RET}(\mathbf{x}^*)$  (separately) considering the three datasets WINE-QUALITY-RED, WINE-QUALITY-WHITE and ONLINE-SHOPPERS. The most precise models and with the best generalization error have been obtained, in

**Table 14**

Evolutionary instance selection methods vs. non-evolutionary instance selection methods: Number of times the algorithm obtains the best value of  $\mathcal{R}_D^{1-NN}(\mathbf{x}^*)$ ,  $\mathcal{R}_D^{C4.5}(\mathbf{x}^*)$ ,  $\mathcal{R}_D^{RF}(\mathbf{x}^*)$ , or  $\mathcal{RET}(\mathbf{x}^*)$ .

Optimization model	$\mathcal{R}_D^{1-NN}(\mathbf{x}^*)$	$\mathcal{R}_D^{C4.5}(\mathbf{x}^*)$	$\mathcal{R}_D^{RF}(\mathbf{x}^*)$	$\mathcal{RET}(\mathbf{x}^*)$
Evolutionary with retention constraint	0	0	0	1
Evolutionary without retention constraint	3	3	3	0
Non-evolutionary	0	0	0	2

the three datasets, with an evolutionary method without retention constraint, although these methods have retained fewer instances than the non-evolutionary methods and, as expected, than the evolutionary methods with retention restriction.

#### 7. A real-life case study: instance selection in a contact center

In this section we turn our attention to real-life data, with two purposes: first, establish how instance selection behaves in real situations, and, second, perform a more in-depth analysis based not only on classification results but also through clustering analysis metrics. The low number of instances of the problem facilitates this deeper analysis. As a matter of fact, the *small data* concept is relevant in many contexts, and in instance selection as well. Following (Kitchin and Lauriault, 2015), we label *small* those datasets generally limited in volume, with non-continuous collection and narrow variety, and that are usually generated to answer specific questions. Small data makes the data accessible, informative and actionable.

The dataset is described below and the results of applying the evolutionary wrapper and filter methods proposed in this paper are analyzed together with the non-evolutionary methods, from both the classification and distance perspectives. Since the application scenario is small data, we have not imposed retention constraint on the evolutionary instance selection methods, and therefore all the evolutionary instance selection methods have been run with  $M = u$ .

##### 7.1. Dataset

The data we have used have been provided by Northern Italy company GAP S.R.L., and consists of the cumulative performances data of 77 agents over a period of 6 months. GAP (Brunello, 2015) is a business process outsourcer operating in various fields, for example, telemarketing and customer care, which offers, among others, inbound and outbound phone-based services. By a complex integration of various data systems, it keeps a rich and detailed record of each agent-user communication, that is, of each session, including both operational and service data. In this dataset, a (subjective) evaluation of the performance in a six-months period (obtained by combining three, independent, expert evaluations) of each agent is associated with a synthesis of the operational and service data generated by their activity in the same period. The aim is to identify the subset of parameters that (implicitly) influence their evaluation, and therefore help the experts in designing a (semi)automatic system for evaluating the agents. The work of the agents has been described via 69 attributes, and, compared to other contact center databases, the quality of the information in this case is considerably higher. The set of variables that describe the agents can be classified into several categories, depending on the particular aspect they describe. The first category is *agent related* variables, and includes their seniority (from 6 months to 5 and an half year), their gender (31 males versus 46 females), their age (from 19 to 65 years old), their level of education (from 1 - minimum compulsory education, to 5 - university degree or more), and their skill average and variance (GAP has internally engineered a skill-function that takes into account several aspects, recomputed weekly for each agent, and of which we consider the average and the variance over the entire period). The

**Table 15**  
Variables describing the agents in the *ALL-AGENTS* dataset.

Agent related variables	
agent_seniority	# of days of service of the agent
agent_gender	whether the agent is a male or female
agent_age	age of the agent
agent_education	level of education of the agent
agent_skill	weekly average and variance of agents' skill
Diversity variables	
num_sessions	daily average and variance of the # of distinct sessions
num_commissions	daily average and variance of the # of distinct commissions
switch_index	daily average and variance of (all) switches
switch_index_flow_type	daily average and variance of flow switches
switch_index_ser_type	daily average and variance of service switches
switch_index_ser_same_type	daily average and variance of sub-service type switches
icc_inbound_av	daily average and variance of avg. icc index in inbound
icc_outbound_av	daily average and variance of avg. icc index in outbound
icc_inbound_var	daily average and variance of var. icc index in inbound
icc_outbound_var	daily average and variance of var. icc index in outbound
Work distribution variables	
management	daily average and variance of # min. working
management_inbound	daily average and variance of # min. working on inbound comm.
management_outbound	daily average and variance of # min. working on outbound comm.
management_backoffice	daily average and variance of # min. working on backoffice
fraction_inbound	daily average and variance of the % of min. on inbound
fraction_outbound	daily average and variance of the % of min. on outbound
fraction_backoffice	daily average and variance of the % of min. on backoffice
available_sessions	daily average and variance of the # of available sessions
available	daily average and variance of # min. available
break_sessions	daily average and variance of the # of break sessions
break	daily average and variance of # min. on break
inactive_sessions	daily average and variance of the # of inactive sessions
inactive	daily average and variance of # min. inactive
Turn distribution variables	
turn_duration	daily average and variance of turn length in # min.
fraction_weekend	fraction of weekend workdays
fraction_night	daily average and variance of the % of min. working during nights
fraction_morning	daily average and variance of the % of min. working during mornings
fraction_early_afternoon	daily average and variance of the % of min. working during early aft.
fraction_late_afternoon	daily average and variance of the % of min. working during late aft.
fraction_evening	daily average and variance of the % of min. working during evening
inactivity_time	fraction of total inactivity time over total turn duration
available_time	fraction of total availability time over total turn duration
break_time	fraction of total break time over total turn duration

second group of variables is agents' *work's diversity*, by means of which it measured how heterogeneous has been the agent's work in the analyzed period; this group includes the number of distinct sessions and distinct commissions the agent has worked on, the daily frequency of context switches, the daily frequency of flow switches (inbound vs. outbound), the daily frequency of service switches, and the daily frequency of sub-type switches. In another group of variables the agents' work distribution is considered, and includes the average and the variance over days of the number of minutes during which he/she has been effectively working, on inbound, on outbound communications, or on backoffice, along with their fraction on the entire workload, that takes into account how many times the agent has declared him/herself available (in idle state), for how many minutes in total, on break, and for how many minutes, and inactive (that is, on break or available). Finally, the last group includes the variables relative to the agents' turns distribution, that take into account in which part of the day and of the week each agent's shifts are mainly scheduled, as well as the fraction, over the entire observed period, of break, available, and inactive time of the agent. In this paper we have named this dataset *ALL-AGENTS*. All variables of *ALL-AGENTS* are listed in Table 15, and Table 16 shows additional information. The dataset has been enriched with a class variable that describes the agent performance value. This value has been obtained by asking to three independent supervisors a fair judgment of each agent to the best of their expertise. Their judgment, on a scale from 1 (lowest) to 5 (highest), takes into account the overall

**Table 16**  
Characteristics of *ALL-AGENTS* dataset.

Dataset	Number of instances	Number of attributes	Attribute type	Number of classes
<i>ALL-AGENTS</i>	77	70	Real, Categorical	2

impression of the agents and their performances; then, the three votes have been combined into a single one by averaging them. This dataset has been previously used in Brunello et al. (2018, 2019).

## 7.2. Classification based analysis

In this section we analyze the predictive capacity of the classifiers trained with the datasets that have been obtained with the proposed instance selection methods and other non-evolutionary method from the literature. The metrics  $ACC_T^{\Phi}(x^*)$ ,  $ACC_D^{\Phi}(x^*)$ ,  $ACC_{CV}^{\Phi}(x^*, 5)$  and  $R_D^{\Phi}(x^*)$ , for  $\Phi = 1 - NN$ ,  $C4.5$ ,  $RF$ , have been used to evaluate those reduced datasets (except  $ACC_T^{1-NN}(x^*)$  because always is 1.0). Table 17 shows accuracy values of the classifiers  $1 - NN$ ,  $C4.5$  and  $RF$  for the *ALL-AGENTS* dataset. Table C.28 report the evaluation of the solution obtained by each of the evolutionary and non-evolutionary instance

selection methods<sup>12</sup> in classifiers 1 – NN, C4.5 and RF. The results for each of the classifiers are analyzed separately below.

- **Results with 1 – NN:** The best 1 – NN classifier has been obtained with the evolutionary method *MODE-W-3o-1NN-ACC-MM-0*, which eliminates 23 instances keeping the same value of the metric  $ACC_D^{1-NN}(x^*)$  and only decreasing by 0.0176 the value of the metric  $ACC_{CV}^{1-NN}(x^*, 5)$ . The best solution found with a non-evolutionary method is obtained with the *PSRCG-TSS* method, which is worse than the solution found by *MODE-W-3o-1NN-ACC-MM-0* in both metrics  $ACC_D^{1-NN}(x^*)$  and  $ACC_{CV}^{1-NN}(x^*, 5)$ , and therefore also in  $R_D^{1-NN}(x^*)$ . Furthermore, the solution of *PSRCG-TSS* removes fewer instances than the *MODE-W-3o-1NN-ACC-MM-0* solution (13 vs. 23). Another aspect to take into account is that the solution of *MODE-W-3o-1NN-ACC-MM-0* is also better than the solution of *PSRCG-TSS* when it is evaluated with the classifiers C4.5 and RF, as can be seen in the values of the metrics  $R_D^{C4.5}(x^*)$  and  $R_D^{RF}(x^*)$  of both solutions. For the sake of illustration, Fig. E.11 shows the Pareto front found with the *MODE-W-3o-1NN-ACC-MM-0* method, in 2 and 3 dimensions, where the red dot marks the solution chosen by the proposed decision making process.
- **Results with C4.5:** The results of this block are important because the algorithm C4.5 builds a decision tree that allows the interpretability of the results of the classification. Once again, *MODE* has been the search strategy that has found the best solution in an wrapper instance selection method with a three-objective optimization model (*MODE-W-3o-C4.5-ACC-R-0* method). However, in this case the evaluator used during the search process is the C4.5 algorithm. This fact was to be expected, that is, if we want to use a decision tree to finally classify our data, the most logical thing is to use decision trees during the instance selection process. This characteristic of being able to choose the classifier during the instance selection process is an advantage that the non-evolutionary methods used in this paper do not have, which are always based on the 1 – NN classifier. In this way, the best decision tree obtained with a non-evolutionary method is found with the *VSM-TSSs0* method, which eliminates fewer instances than the *MODE-W-3o-C4.5-ACC-R-0* method, and the accuracy of the obtained decision tree is worse than the precision of the decision tree obtained with the method *MODE-W-3o-C4.5-ACC-R-0* in all metrics analyzed. Fig. E.12 shows the Pareto front found with the *MODE-W-3o-C4.5-ACC-R-0* method.
- **Results with RF:** Finally, we analyze the results of instance selection methods when they are evaluated using random forest classifiers. The use of RF to evaluate instance selection methods is also significant since these classifiers are very accurate, and as shown in Table 17, it achieves better classification results than 1 – NN and C4.5 for the *ALL-AGENTS* dataset. In this case, the best result has been obtained with the filter-type evolutionary instance selection method proposed in this paper, together with the *NSGA-II* search strategy and the *Min–Max* constraint handling technique (*NSGA-II-F-3o-CON-COR-MM-0* method). This demonstrates the good performance of the proposed filter evolutionary instance selection method based on correlation-redundancy and consistency, which, in this case, beats the wrapper evolutionary instance selection methods even when RF is used as evaluator in the optimization process. The solution obtained with the *NSGA-II-F-3o-CON-COR-MM-0* method, although it degrades the  $ACC_D^{RF}(x^*)$  metric by 0.026, nevertheless improves the  $ACC_{CV}^{RF}(x^*, 5)$  metric by 0.0176 with respect to the original dataset, eliminating 4 instances. Fig. E.13 shows the Pareto front found with the *NSGA-II-F-3o-CON-COR-MM-0* method.

<sup>12</sup> The results of the *POP-TSS* method have been struck out because no instances have been removed.

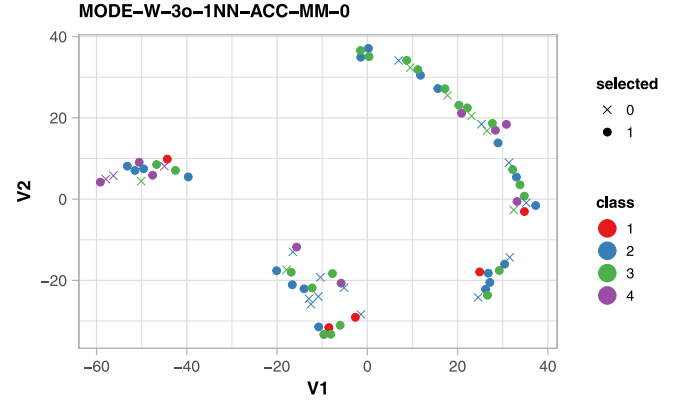


Fig. 2. *MODE-W-3o-1NN-ACC-MM-0* removed instances distribution.

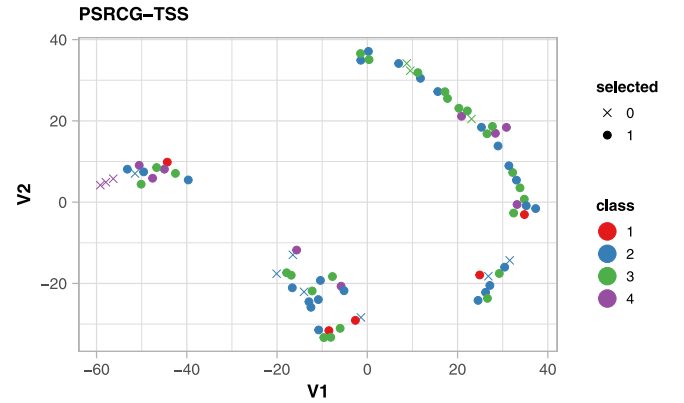


Fig. 3. *PSRCG-TSS* removed instances distribution.

We can conclude that the wrapper instance selection method with the *MODE* search strategy together with the three-objective optimization model proposed in this paper (*MODE-W-3o-1NN-ACC-MM-0* method), and the filter instance selection method with the *NSGA-II* search strategy together with the correlation-redundancy and consistency based optimization model (*NSGA-II-F-3o-CON-COR-MM-0* method) are the best alternatives to reduce the set of instances of the *ALL-AGENTS* dataset, or the *MODE-W-3o-C4.5-ACC-R-0* method if what we are looking for is an interpretable classifier.

### 7.3. Distance-based analysis

In this section, the best instance selection methods are analyzed and compared from a distance-based perspective. Since the best non-evolutionary method (*PSRCG-TSS*) is based on nearest neighbors classification, for a fair comparison we have only compared it with the best of the evolutionary methods when it is evaluated with the 1 – NN classifier (*MODE-W-3o-1NN-ACC-MM-0*). We want to analyze how classes distributions have been affected by the instance selection process. Before analyzing the instance selection methods, we want to visually analyze how removed instances are distributed. To this end, *t-distributed Stochastic Neighbor Embedding* (*t-SNE*) has been used (Van der Maaten and Hinton, 2008). *t-SNE* allows the visualization of high-dimensional data by assigning each instance a location in a two-dimensional or three-dimensional space.

Figs. 2 and 3 depicts how removed instances are distributed. As can be seen, in most cases, the removed instances are close to the retained



**Table 17**Accuracy values of 1 – NN, C4.5 and Random Forest classifiers using the complete dataset  $D$  (solution  $\mathbf{x}_D$ ) of ALL-AGENTS.

Dataset	$ACC_D^{1-NN}(\mathbf{x}_D)$	$ACC_{CV}^{1-NN}(\mathbf{x}_D, 5)$	$ACC_D^{C4.5}(\mathbf{x}_D)$	$ACC_{CV}^{C4.5}(\mathbf{x}_D, 5)$	$ACC_D^{RF}(\mathbf{x}_D)$	$ACC_{CV}^{RF}(\mathbf{x}_D, 5)$
ALL-AGENTS	1.0000	0.4805	0.9221	0.5195	1.0000	0.5714

**Table 18**

Distance-based measures for different instance selection methods and original ALL-AGENTS dataset.

Dataset/Method	$C(\mathbf{x}^*)$	Maximum diameter	Minimum separation	$\Psi_D(\mathbf{x}^*)$
ALL-AGENTS	77	24.908	2.960	–
MODE-W-3o-1NN-ACC-MM-0	54	<b>24.061</b>	2.978	<b>6.496</b>
PSRCG-TSS	64	24.807	<b>3.038</b>	6.983

instances, keeping at least one class representative per group. Nevertheless, to complete this subjective initial analysis, a more objective analysis has been performed based on distances measures.

Table 18 shows the results of different distance measures applied to the original dataset ALL-AGENTS and the reduced datasets obtained from different instance selection methods. *Maximum diameter* stands for the maximum distance between two instances of the same class and *minimum separation* measures the minimum distance between two instances of different classes. The last column presents the *retained-removed sample proximity* metric proposed in this paper, which includes selected and removed instances, unlike the previous ones. Given a solution  $\mathbf{x}^*$  obtained with an instance selection method, and the solution  $\mathbf{x}_D$  with all instances of  $D$ , the retained-removed sample proximity metric  $\Psi_D(\mathbf{x}^*)$  calculates the mean of the normalized distances between each instance of  $\mathbf{x}^*$  and the nearest neighbor of its same class in  $\mathbf{x}_D - \mathbf{x}^*$ . Function  $\Psi_D$  is formulated as follows:

$$\Psi_D(\mathbf{x}) = \frac{1}{C(\mathbf{x})} \sum_{\substack{i=1, \dots, w \\ x_j=1}} NN_D(i, \mathbf{x}_D - \mathbf{x}) \quad (8)$$

$$NN_D(i, \mathbf{z}) = \min_{\substack{j=1, \dots, w \\ z_j=1}} d(I_i, I_j) \quad (9)$$

$$d(I_i, I_j) = \frac{1}{n} \sum_{k=1, \dots, n} \sqrt{\left( \frac{a_k^i - a_k^j}{a_k^{max} - a_k^{min}} \right)^2} \quad (10)$$

where  $a_k^{max} = \max_{i=1, \dots, w} a_k^i$  and  $a_k^{min} = \min_{i=1, \dots, w} a_k^i$ . According with the data depicted in Table 18, we can conclude that the solution obtained with the evolutionary method MODE-W-3o-1NN-ACC-MM-0 is better than the solution obtained with the non-evolutionary method PSRCG-TSS. In addition to eliminating more instances, the evolutionary method obtains better values for the maximum diameter and retained-removed sample proximity metrics, presenting very similar values in the minimum separation metric. In this sense, the evolutionary method produces more compact class distributions as well as maintains class separations.

## 8. Conclusions and future work

Instance selection is a very important process within prediction systems and this is confirmed by the large number of works that currently exist in the state of the art. Instance selection can be required in two scenarios: (1) you want to eliminate irrelevant, inconsistent or redundant instances for more compact and accurate models, and (2) you want to select a percentage of instances of a dataset because current computer systems for machine learning do not support such a large amount of data. Most of today's works are in the first scenario, and cannot deal with problems in the second scenario due to the enormous amount of memory required and processing time. The approaches proposed in this paper deal with problems of both types of scenarios.

For second scenario problems, the size of the selected set of instances is controlled through constraints imposed on the optimization problem, which are set by the user. In this way, the sets of instances with a number higher than the one set by the user (infeasible solutions) are not evaluated but are simply repaired or evaluated only through the constraint violations (according to the implemented constraint handling method) thus avoiding the high computational cost (memory and run time) that this would entail.

In addition to the advantages discussed in the field of big data, our proposal also has advantages in the field of machine learning. The third objective optimized with the wrapper methods proposed in this paper helps to improve the generalization error, as confirmed by the experiments carried out. Regarding the filter methods, appropriate in the presence of large amounts of data, the simultaneous optimization of correlation + redundancy together with the consistency of the data has produced an acceptable Pareto front, for the datasets analyzed, on which to choose the final solution, in reasonable run times.

As main future works we are considering extending the proposed instance selection methods to regression, time series and cost sensitive classification problems. Furthermore, we are already working on a surrogate-assisted evolutionary algorithm to reduce the computational time required by wrapper instance selection methods, especially when the learning algorithm is very time consuming, such as deep learning. Finally, we are also already working on a multi-objective filter instance selection method where one of the objectives is to minimize the retained-removed sample proximity measure proposed in this paper.

## CRedit authorship contribution statement

**Fernando Jiménez:** Term, Conceptualization, Formal analysis, Methodology, Software, Validation, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Supervision, Visualization, Project administration, Funding acquisition. **Gracia Sánchez:** Conceptualization, Formal analysis, Methodology, Software, Investigation, Validation, Data curation, Visualization. **José Palma:** Methodology, Formal analysis, Writing – original draft, Writing – review & editing, Funding acquisition. **Guido Sciavicco:** Validation, Data curation, Writing – original draft, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was partially funded by the SITSUS project (Ref: RTI2018-094832-B-I00), given by the Spanish Ministry of Science, Innovation and Universities (MCIU), the Spanish Agency for Research (AEI) and by the European Regional Development Fund (FEDER). This work was supported by the Science and Technology Agency, Séneca Foundation, Comunidad Autónoma de la Región de Murcia, Spain through the research projects 00004/COVI/20 and 00007/COVI/20.

## Appendix A. Algorithms

---

### Algorithm 1 NSGA-II

---

**Require:**  $T > 1$  {Number of generations}  
**Require:**  $N > 1$  {Number of individuals in the population}  
1:  $P \leftarrow \text{InitializePopulation}(N)$   
2: **for**  $I \in P$  **do**  
3:    $\text{Evaluate}(I)$   
4: **end for**  
5:  $t \leftarrow 0$   
6: **while**  $t < T$  **do**  
7:    $Q \leftarrow \emptyset$   
8:    $i \leftarrow 0$   
9:   **while**  $i < N$  **do**  
10:      $\text{Parent1} \leftarrow \text{BinaryTournamentSelection}(P)$   
11:      $\text{Parent2} \leftarrow \text{BinaryTournamentSelection}(P)$   
12:      $(\text{Offspring1}, \text{Offspring2}) \leftarrow \text{EAVariation}(\text{Parent1}, \text{Parent2})$   
13:      $\text{Evaluate}(\text{Offspring1})$   
14:      $\text{Evaluate}(\text{Offspring2})$   
15:      $Q \leftarrow Q \cup \{\text{Offspring1}, \text{Offspring2}\}$   
16:      $i \leftarrow i + 2$   
17:   **end while**  
18:    $R \leftarrow P \cup Q$   
19:    $P \leftarrow N$  best individuals in  $R$  according to the *RankCrowding* order relation  
20:    $t \leftarrow t + 1$   
21: **end while**  
22: **return** Non-dominated individuals in population  $P$

---



---

### Algorithm 2 BinaryTournamentSelection

---

**Require:**  $S$  {Set of individuals}  
1:  $I \leftarrow$  Random selection from  $S$   
2:  $J \leftarrow$  Random selection from  $S$   
3: **if**  $I$  is better than  $J$  in  $S$  according to the *RankCrowding* order relation **then**  
4:   **return**  $I$   
5: **else**  
6:   **return**  $J$   
7: **end if**

---



---

### Algorithm 3 RankCrowding

---

**Require:**  $S$  {Set of individuals}  
**Require:**  $I \in S, J \in S$  {Individuals to compare in  $S$ }  
1: **if**  $\text{rank}(S, I) < \text{rank}(S, J)$  **then**  
2:   **return** *True*  
3: **end if**  
4: **if**  $\text{rank}(S, I) > \text{rank}(S, J)$  **then**  
5:   **return** *False*  
6: **end if**  
7:  $F_I \leftarrow \{K \in S \mid \text{rank}(P, K) = \text{rank}(P, I)\}$   
8:  $F_J \leftarrow \{K \in S \mid \text{rank}(S, K) = \text{rank}(S, J)\}$   
9: **return**  $\text{CrowdingDistance}(F_I, I) > \text{CrowdingDistance}(F_J, J)$

---



---

### Algorithm 4 CrowdingDistance

---

**Require:**  $S$  {Set of individuals}  
**Require:**  $I \in S$  {Individual in  $S$ }  
**Require:**  $l$  {Number of objectives}  
1: **for**  $j = 1$  to  $l$  **do**  
2:    $\max_j \leftarrow \max_{I \in S} \{f_j(I)\}$   
3:    $\min_j \leftarrow \min_{I \in S} \{f_j(I)\}$   
4:    $\text{suc}_j \leftarrow$  Individual higher adjacent to the individual  $I$  in the  $j$ th objective  
5:    $\text{pre}_j \leftarrow$  Individual lower adjacent to the individual  $I$  in the  $j$ th objective  
6: **end for**  
7: **for**  $j = 1$  to  $l$  **do**  
8:   **if**  $f_j(I) = \max_j$  or  $f_j(I) = \min_j$  **then**  
9:     **return**  $\infty$   
10:   **end if**  
11: **end for**  
12:  $CD \leftarrow 0.0$   
13: **for**  $j = 1$  to  $l$  **do**  
14:    $CD \leftarrow CD + \frac{f_j(\text{suc}_j) - f_j(\text{pre}_j)}{\max_j - \min_j}$   
15: **end for**  
16: **return**  $CD$

---



---

### Algorithm 5 MODE

---

**Require:**  $T > 1$  {Number of generations}  
**Require:**  $N \geq 1$  {Number of individuals in the population}  
1:  $P \leftarrow \text{InitializePopulation}(N)$   
2: **for**  $I \in P$  **do**  
3:    $\text{Evaluate}(I)$   
4: **end for**  
5:  $t \leftarrow 0$   
6: **while**  $t < T$  **do**  
7:    $Q \leftarrow \emptyset$   
8:    $i \leftarrow 0$   
9:   **for**  $I \in P$  **do**  
10:      $\text{Offspring} \leftarrow \text{DEVariation}(I, P)$   
11:      $\text{Evaluate}(\text{Offspring})$   
12:      $Q \leftarrow Q \cup \{\text{Offspring}\}$   
13:   **end for**  
14:    $R \leftarrow P \cup Q$   
15:    $P \leftarrow N$  best individuals in  $R$  according to the *RankCrowding* order relation  
16:    $t \leftarrow t + 1$   
17: **end while**  
18: **return** Non-dominated individuals in population  $P$

---

**Algorithm 6** *InitializePopulation*


---

**Require:**  $\delta > 0$  {Number of crossover operators}  
**Require:**  $\epsilon > 0$  {Number of mutation operators}  
**Require:**  $w$  {Number of instances in dataset}  
**Require:**  $N \geq 0$  {Number of individuals in the population}

```

1:  $P \leftarrow \emptyset$ 
2: while  $|P| < N$  do
3:    $Q \leftarrow \{1, \dots, w\}$ 
4:    $q \leftarrow$  Random integer from  $Q$ 
5:    $r \leftarrow w - q$ 
6:   for  $i = 1$  to  $q$  do
7:      $j \leftarrow$  Random integer from  $Q$ 
8:      $b_j^I \leftarrow 1$  {instance  $j$  of individual  $I$  is selected}
9:      $Q \leftarrow Q - \{j\}$ 
10:  end for
11:  for  $i = 1$  to  $r$  do
12:     $j \leftarrow$  Random integer from  $Q$ 
13:     $b_j^I \leftarrow 0$  {instance  $j$  of individual  $I$  is not selected}
14:     $Q \leftarrow Q - \{j\}$ 
15:  end for
16:   $d_I \leftarrow$  Random integer from  $\{0, \dots, \delta\}$  {only in NSGA-II}
17:   $e_I \leftarrow$  Random integer from  $\{0, \dots, \epsilon\}$  {only in NSGA-II}
18:   $m_I \leftarrow$  Random real number from  $[0, 1]$  {only in NSGA-II}
19:   $c_I \leftarrow$  Random real number from  $[0, 1]$  {only in MODE}
20:   $I \leftarrow \text{Repair}(I)$  {only in constraint handling based on repair method}
21:   $P \leftarrow P \cup \{I\}$  {Add the individual  $I$  to the population  $P$ }
22: end while
23: return  $P$ 
```

---

**Algorithm 7** *Repair*


---

**Require:**  $I$  {Individual to repair}

```

1:  $\mathbf{x} \leftarrow \{b_1^I, \dots, b_w^I\}$ 
2: while  $Q_D(\mathbf{x}_D) < Q_D(\mathbf{x})$  do
3:   Add to  $\mathbf{x}$  a random instance with a class missing in  $\mathbf{x}$ 
4: end while
5: while  $C(\mathbf{x}) < p$  do
6:   Add to  $\mathbf{x}$  a random instance with the class with fewer instances in  $\mathbf{x}$ 
7: end while
8: while  $C(\mathbf{x}) > M$  do
9:   Remove from  $\mathbf{x}$  a random instance with the class with more instances in  $\mathbf{x}$ 
10: end while
11: return  $\mathbf{x}$ 
```

---

**Algorithm 8** *Evaluate* (constraint handling based on repair method)

---

**Require:**  $I$  {Individual to evaluate}  
**Require:**  $D$  {Dataset}  
**Require:**  $\Phi$  {Learning algorithm, only for problem (2)}  
**Require:**  $p$  {Number of folds for cross-validation, only for problem (2)}

```

1:  $\mathbf{x} \leftarrow \{b_1^I, \dots, b_w^I\}$ 
2:  $F_I \leftarrow \{F_D^\Phi(\mathbf{x}), F_{CV}^\Phi(\mathbf{x}, p), C(\mathbf{x})\}$  {in problem (2)}
3:  $F_I \leftarrow \{C\mathcal{O}\mathcal{N}_D(\mathbf{x}), C\mathcal{O}\mathcal{R}_D(\mathbf{x}), C(\mathbf{x})\}$  {in problem (3)}
```

---

**Algorithm 9** *Evaluate* (constraint handling based on  $Min - Max$  criterion)

---

**Require:**  $I$  {Individual to evaluate}  
**Require:**  $D$  {Dataset}  
**Require:**  $\Phi$  {Learning algorithm, only for problem (2)}  
**Require:**  $p$  {Number of folds for cross-validation, only for problem (2)}

```

1:  $\mathbf{x} \leftarrow \{b_1^I, \dots, b_w^I\}$ 
2: if  $g(\mathbf{x}) > 0$  then
3:    $F \leftarrow g(\mathbf{x})$ 
4: else
5:    $F \leftarrow \{F_D^\Phi(\mathbf{x}), F_{CV}^\Phi(\mathbf{x}, p), C(\mathbf{x})\}$  {in problem (2)}
6:    $F \leftarrow \{C\mathcal{O}\mathcal{N}_D(\mathbf{x}), C\mathcal{O}\mathcal{R}_D(\mathbf{x}), C(\mathbf{x})\}$  {in problem (3)}
7: end if
8: return  $F$ 
```

---

**Algorithm 10** *EAVariation* (Variation in Evolutionary Algorithms)

---

**Require:**  $Parent1, Parent2$  {Individuals to vary}

```

1:  $(Child1, Child2) \leftarrow \text{EASelfAdaptiveCrossover}(Parent1, Parent2)$ 
2:  $Offspring1 \leftarrow \text{EASelfAdaptiveMutation}(Child1)$ 
3:  $Offspring2 \leftarrow \text{EASelfAdaptiveMutation}(Child2)$ 
4:  $Offspring1 \leftarrow \text{Repair}(Offspring1)$  {only in constraint handling based on repair method}
5:  $Offspring2 \leftarrow \text{Repair}(Offspring2)$  {only in constraint handling based on repair method}
6: return  $(Offspring1, Offspring2)$ 
```

---

**Algorithm 11** *EASelfAdaptiveCrossover* (Self-adaptive crossover in Evolutionary Algorithms)

---

**Require:**  $I, J$  {Individuals for crossover}  
**Require:**  $p_v$  ( $0 < p_v < 1$ ) {Probability of operator change}  
**Require:**  $\delta > 0$  {Number of different crossover operators ( $\delta = 1$  in our case)}

```

1:  $K \leftarrow I$ 
2:  $L \leftarrow J$ 
3: if a Bernoulli random variable takes the value 1 with probability  $p_v$  then
4:    $d_K \leftarrow$  Random integer from  $[0, \delta]$ 
5: end if
6:  $d_L \leftarrow d_K$ 
7: if  $d_K = 1$  then
8:    $(K, L) \leftarrow \text{EAUniformCrossover}(K, L)$ 
9: end if
10: return  $(K, L)$ 
```

---



**Algorithm 12** *EAUniformCrossover* (Uniform crossover in Evolutionary Algorithm)

---

**Require:**  $I, J$  {Individuals for crossover}

- 1:  $K \leftarrow I$
- 2:  $L \leftarrow J$
- 3: **for**  $i = 1$  **to**  $w$  **do**
- 4:   **if** a Bernoulli random variable takes the value 1 with probability 0.5 **then**
- 5:      $b_i^K \leftarrow b_i^J$
- 6:      $b_i^L \leftarrow b_i^I$
- 7:   **end if**
- 8: **end for**
- 9: **return**  $(K, L)$

---

**Algorithm 13** *EASelfAdaptiveMutation* (Self-adaptive mutation in Evolutionary Algorithms)

---

**Require:**  $I$  {Individual to mutate}

**Require:**  $p_v$  ( $0 < p_v < 1$ ) {Probability of operator change}

**Require:**  $\epsilon > 0$  {Number of different mutation operators ( $\epsilon = 1$  in our case)}

- 1:  $J \leftarrow I$
- 2: **if** a Bernoulli random variable takes the value 1 with probability  $p_v$  **then**
- 3:    $e_J \leftarrow$  Random integer from  $[0, \epsilon]$
- 4: **end if**
- 5: **if**  $e_J = 1$  **then**
- 6:    $J \leftarrow$  *EASelfAdaptiveBitFlipMutation*( $J$ )
- 7: **end if**
- 8: **return**  $J$

---

**Algorithm 14** *EASelfAdaptiveBitFlipMutation* (Self-adaptive bit-flip mutation in Evolutionary Algorithms)

---

**Require:**  $I$  {Individual to mutate}

**Require:**  $p_v$  ( $0 < p_v < 1$ ) {Probability of change in mutation probability}

- 1:  $J \leftarrow I$
- 2: **if** a Bernoulli random variable takes the value 1 with probability  $p_v$  **then**
- 3:    $m_J \leftarrow$  Random real number from  $[0, 1]$
- 4: **end if**
- 5: **for**  $i = 1$  **to**  $w$  **do**
- 6:   **if** a Bernoulli random variable takes the value 1 with probability  $m_J$  **then**
- 7:     **if**  $b_i^J = 1$  **then**
- 8:        $b_i^J \leftarrow 0$
- 9:     **else**
- 10:        $b_i^J \leftarrow 1$
- 11:     **end if**
- 12:   **end if**
- 13: **end for**
- 14: **return**  $J$

---

**Algorithm 15** *DEVariation* (Variation in Differential Evolution)

---

**Require:**  $Parent$  {Individual to vary}

**Require:**  $S$  {Set of individuals}

- 1:  $Mutant \leftarrow$  *DEBinaryDifferentialMutation*( $Parent, S$ )
- 2:  $Offspring \leftarrow$  *DESelfAdaptiveBinomialCrossover*( $Parent, Mutant$ )
- 3:  $Offspring \leftarrow$  *Repair*( $Offspring$ ) {only in constraint handling based on repair method}
- 4: **return**  $Offspring$

---

**Algorithm 16** *DEBinaryDifferentialMutation* (Binary mutation in Differential Evolution)

---

**Require:**  $I$  {Individual to mutate}

**Require:**  $S$  {Set of individuals}

- 1:  $J \leftarrow I$
- 2:  $A \leftarrow$  Random non-dominated individual from  $S$
- 3:  $B \leftarrow$  Random individual from  $S$  distinct from  $A$  and  $J$
- 4:  $C \leftarrow$  Random individual from  $S$  distinct from  $A, B$  and  $J$
- 5: **for**  $i = 1$  **to**  $w$  **do**
- 6:   **if**  $b_i^B = b_i^C$  **then**
- 7:      $b_i^J \leftarrow b_i^A$
- 8:   **else**
- 9:      $b_i^J \leftarrow b_i^B$
- 10:   **end if**
- 11: **end for**
- 12: **return**  $J$

---

**Algorithm 17** *DESelfAdaptiveBinomialCrossover* (Self-adaptive crossover in Differential Evolution)

---

**Require:**  $I, J$  {Individuals for crossover}

**Require:**  $p_v$  ( $0 < p_v < 1$ ) {Probability of change in crossover probability}

- 1:  $K \leftarrow I$
- 2: **if** a Bernoulli random variable takes the value 1 with probability  $p_v$  **then**
- 3:    $c_K \leftarrow$  Random real number from  $[0, 1]$
- 4: **end if**
- 5: **for**  $i = 1$  **to**  $w$  **do**
- 6:   **if** a Bernoulli random variable takes the value 1 with probability  $c_K$  **then**
- 7:      $b_i^K \leftarrow b_i^J$
- 8:   **end if**
- 9: **end for**
- 10: **return**  $K$

---

## Appendix B. Organization charts

See Figs. B.4–B.7.

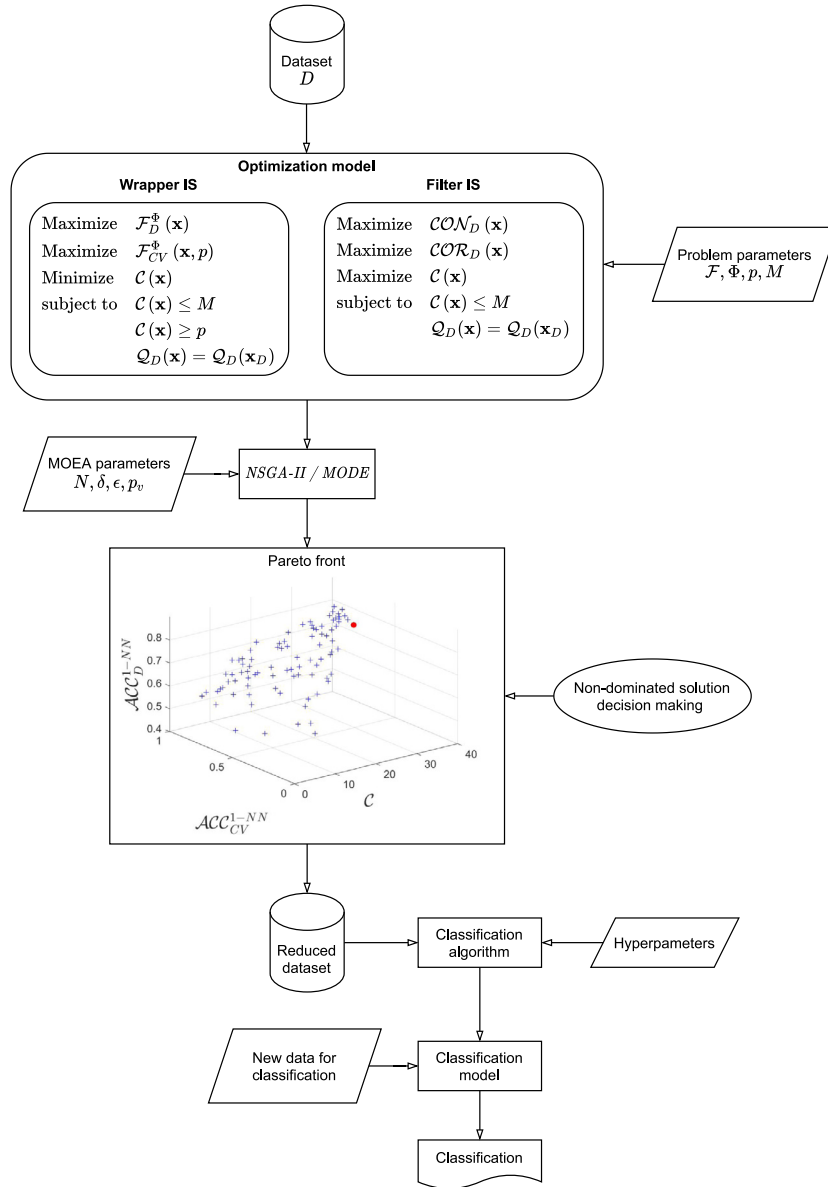


Fig. B.4. Instance selection based classification process proposed in this paper.

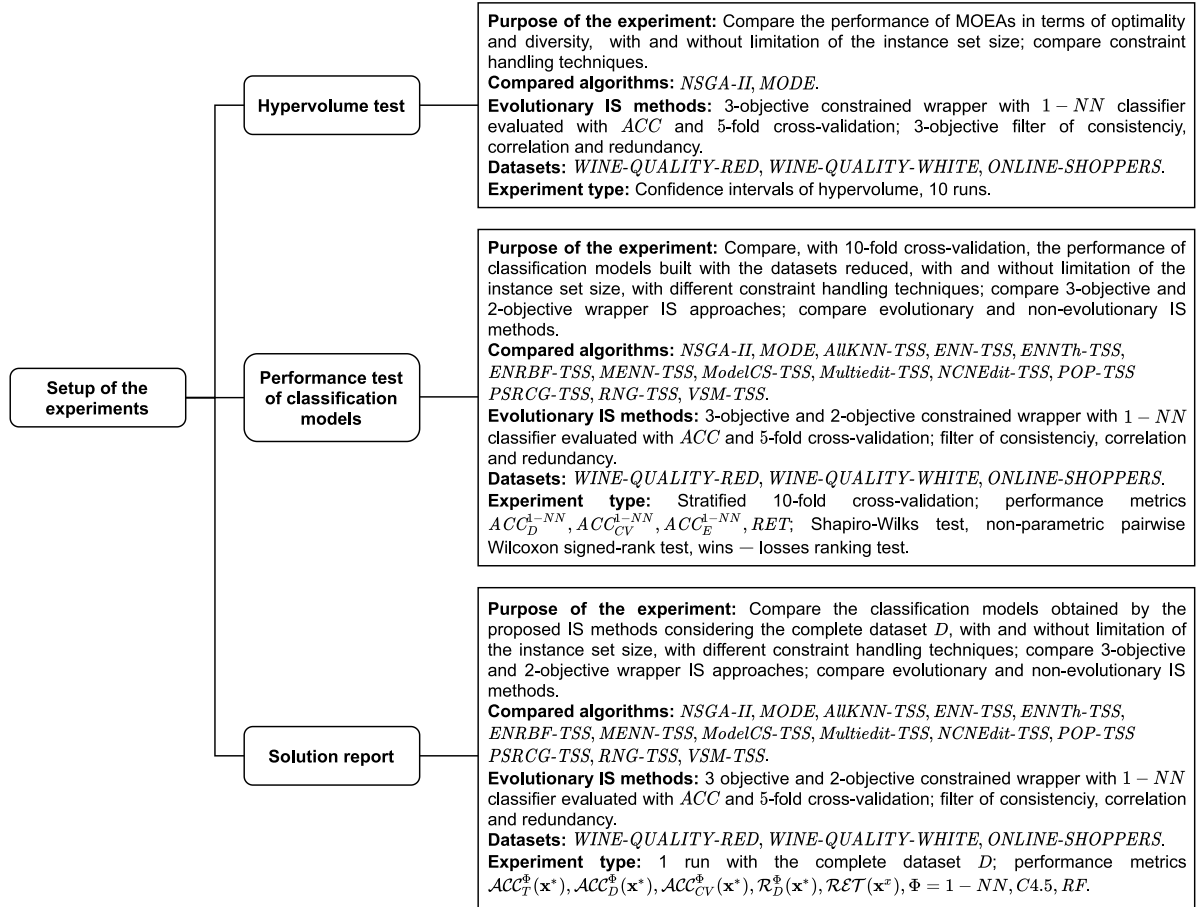


Fig. B.5. Setup of the experiments.



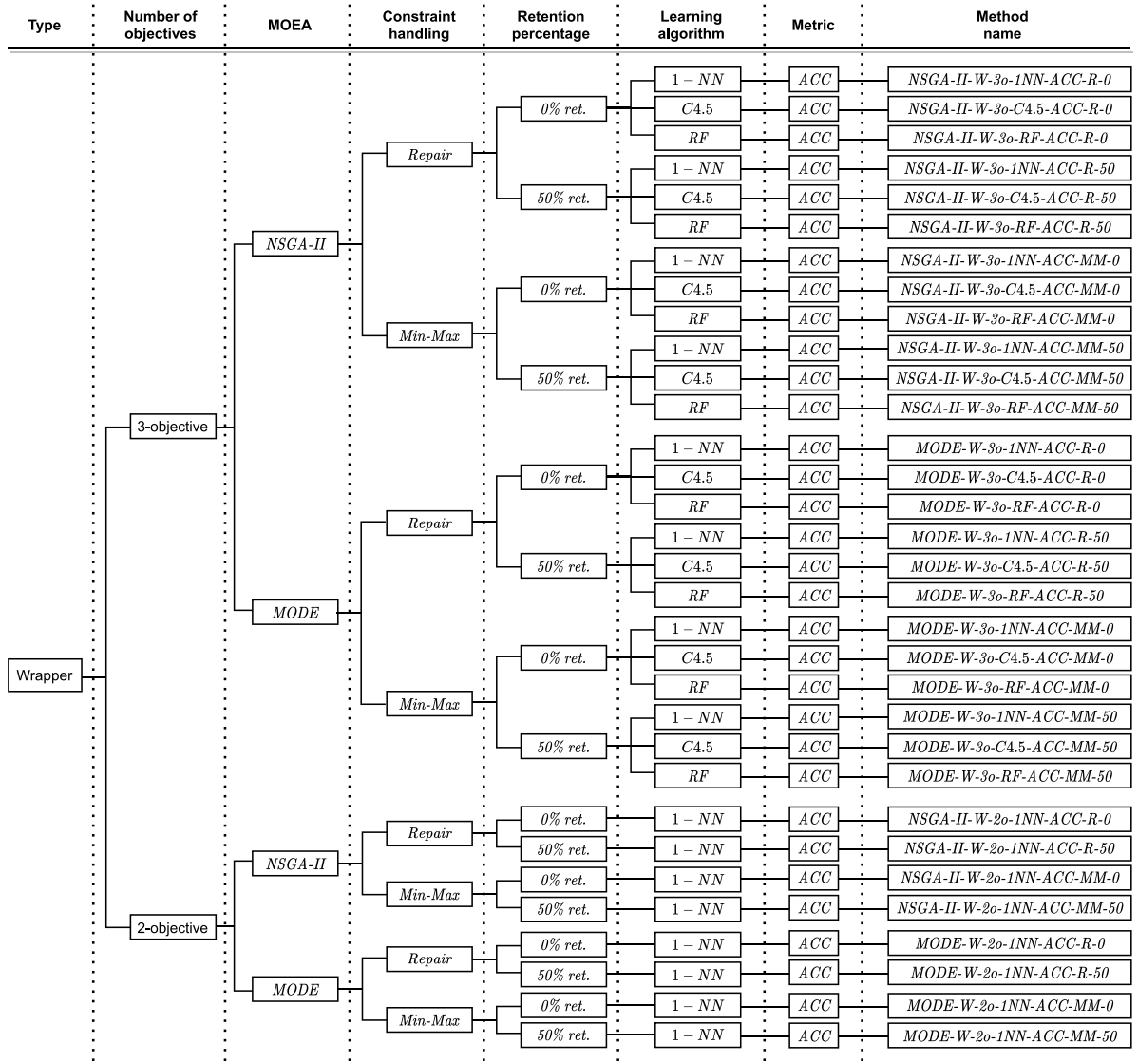


Fig. B.6. Evolutionary wrapper instance selection methods used in this paper.

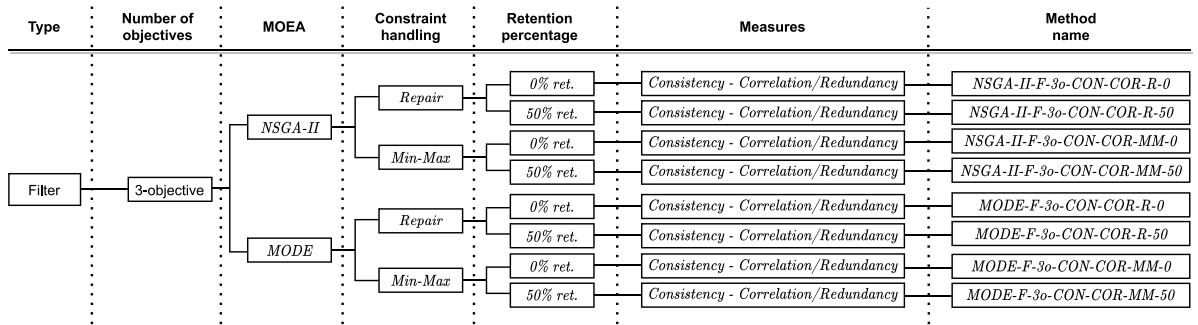


Fig. B.7. Evolutionary filter instance selection methods used in this paper.

## Appendix C. Results tables

See Tables C.19–C.28.

**Table C.19**  
Results of stratified 10-fold cross-validation for *WINE-QUALITY-RED*.

Method	$\overline{ACC}_D^{1-NN}$	$\overline{ACC}_{CV}^{1-NN}$	$\overline{ACC}_E^{1-NN}$	$\overline{RET}$
<i>NSGA-II-W-3o-1NN-ACC-R-0</i>	0.9960	0.6178	0.6505	98.6381
<i>MODE-W-3o-1NN-ACC-R-0</i>	0.9976	0.6179	0.6524	98.6382
<i>NSGA-II-F-3o-CON-COR-R-0</i>	0.9952	0.6211	0.6474	99.0135
<i>MODE-F-3o-CON-COR-R-0</i>	0.9967	0.6191	0.6499	99.3191
<i>NSGA-II-W-2o-1NN-ACC-R-0</i>	0.9672	0.4599	0.6416	70.0874
<i>MODE-W-2o-1NN-ACC-R-0</i>	0.9929	0.5588	0.6486	86.7672
<i>NSGA-II-W-3o-1NN-ACC-MM-0</i>	0.9983	0.6256	0.6524	98.5752
<i>MODE-W-3o-1NN-ACC-MM-0</i>	0.9981	0.6222	0.6537	99.2147
<i>NSGA-II-F-3o-CON-COR-MM-0</i>	0.9979	0.6221	0.6530	99.7151
<i>MODE-F-3o-CON-COR-MM-0</i>	0.9983	0.6201	0.6518	99.6594
<i>NSGA-II-W-2o-1NN-ACC-MM-0</i>	0.9651	0.5109	0.6394	72.7551
<i>MODE-W-2o-1NN-ACC-MM-0</i>	0.9978	0.5608	0.6505	88.9650
<hr/>				
<i>NSGA-II-W-3o-1NN-ACC-R-50</i>	0.8199	0.5087	0.5873	44.9593
<i>MODE-W-3o-1NN-ACC-R-50</i>	0.8482	0.5700	0.5967	49.8436
<i>NSGA-II-F-3o-CON-COR-R-50</i>	0.7564	0.6500	0.5855	49.6560
<i>MODE-F-3o-CON-COR-R-50</i>	0.7181	0.7420	0.5898	49.8229
<i>NSGA-II-W-2o-1NN-ACC-R-50</i>	0.8024	0.3847	0.5929	28.9722
<i>MODE-W-2o-1NN-ACC-R-50</i>	0.8694	0.4640	0.6062	41.5362
<i>NSGA-II-W-3o-1NN-ACC-MM-50</i>	0.8674	0.4979	0.5943	47.4600
<i>MODE-W-3o-1NN-ACC-MM-50</i>	0.8444	0.5860	0.6085	49.6144
<i>NSGA-II-F-3o-CON-COR-MM-50</i>	0.7138	0.7393	0.5731	49.7881
<i>MODE-F-3o-CON-COR-MM-50</i>	0.7300	0.7185	0.5884	49.8853
<i>NSGA-II-W-2o-1NN-ACC-MM-50</i>	0.8441	0.4564	0.6036	35.4408
<i>MODE-W-2o-1NN-ACC-MM-50</i>	0.8718	0.4804	0.6006	44.2554
<hr/>				
<i>AIKNN-TSS</i>	0.5904	0.9147	0.5430	25.8461
<i>ENN-TSS</i>	0.6049	0.9037	0.4808	42.6033
<i>ENNTh-TSS</i>	0.4866	0.7099	0.4619	12.5928
<i>ENRBF-TSS</i>	0.6335	0.9147	0.5362	49.6277
<i>MENN-TSS</i>	0.4459	0.7246	0.4375	10.1238
<i>ModelCS-TSS</i>	0.7807	0.6671	0.4925	65.3637
<i>Multiedit-TSS</i>	0.4635	0.9576	0.3611	26.3172
<i>NCNEdit-TSS</i>	0.6322	0.8251	0.5077	43.3131
<i>POP-TSS</i>	1.0000	0.4360	0.4626	100.0000
<i>PSRCG-TSS</i>	0.9798	0.3738	0.4440	88.7390
<i>RNG-TSS</i>	0.6349	0.8756	0.5111	44.8574
<i>VSM-TSSs0</i>	0.9276	0.3751	0.4400	82.8093

**Table C.20**  
Results of stratified 10-fold cross-validation for *WINE-QUALITY-WHITE*.

Method	$\overline{ACC}_D^{1-NN}$	$\overline{ACC}_{CV}^{1-NN}$	$\overline{ACC}_E^{1-NN}$	$\overline{RET}$
<i>NSGA-II-F-3o-CON-COR-R-0</i>	0.9970	0.6255	0.6722	99.2218
<i>MODE-F-3o-CON-COR-R-0</i>	0.9973	0.6255	0.6724	99.2717
<i>NSGA-II-F-3o-CON-COR-MM-0</i>	0.9963	0.6240	0.6708	99.1290
<i>MODE-F-3o-CON-COR-MM-0</i>	0.9961	0.6252	0.6722	99.0247
<hr/>				
<i>NSGA-II-F-3o-CON-COR-R-50</i>	0.7688	0.5987	0.5819	49.9024
<i>MODE-F-3o-CON-COR-R-50</i>	0.7493	0.6350	0.5813	49.9229
<i>NSGA-II-F-3o-CON-COR-MM-50</i>	0.7552	0.6180	0.5765	49.9728
<i>MODE-F-3o-CON-COR-MM-50</i>	0.7567	0.6172	0.5832	49.9342
<hr/>				
<i>AIKNN-TSS</i>	0.6642	0.9235	0.5805	38.4340
<i>ENN-TSS</i>	0.6984	0.8933	0.5980	52.5748
<i>ENNTh-TSS</i>	0.6302	0.9452	0.5749	22.4590
<i>ENRBF-TSS</i>	0.5005	0.9827	0.4928	47.8077
<i>MENN-TSS</i>	0.6298	0.9506	0.573	22.4103
<i>ModelCS-TSS</i>	0.8658	0.7655	0.6337	76.4995
<i>Multiedit-TSS</i>	0.6251	0.9279	0.5723	42.9363
<i>NCNEdit-TSS</i>	0.7674	0.8724	0.5917	64.5202
<i>POP-TSS</i>	1.0000	0.6142	0.6518	100.0000
<i>PSRCG-TSS</i>	0.9767	0.4283	0.6305	73.2885
<i>RNG-TSS</i>	0.7341	0.8816	0.6137	58.8428
<i>VSM-TSSs0</i>	0.8847	0.5024	0.5898	69.3141

Table C.21

Results of stratified 10-fold cross-validation for *ONLINE-SHOPPERS*.

Method	$\overline{ACC}_D^{1-NN}$	$\overline{ACC}_{CV}^{1-NN}$	$\overline{ACC}_E^{1-NN}$	$\overline{RET}$
<i>NSGA-II-F-3o-CON-COR-R-0</i>	0.9990	0.8123	0.8138	99.4196
<i>MODE-F-3o-CON-COR-R-0</i>	0.9991	0.8120	0.8135	99.4917
<i>NSGA-II-F-3o-CON-COR-MM-0</i>	0.9983	0.8135	0.8142	99.2286
<i>MODE-F-3o-CON-COR-MM-0</i>	0.9987	0.8133	0.8136	99.2854
<i>NSGA-II-F-3o-CON-COR-R-50</i>	0.9036	0.8088	0.8101	49.9946
<i>MODE-F-3o-CON-COR-R-50</i>	0.8881	0.8384	0.8122	49.8964
<i>NSGA-II-F-3o-CON-COR-MM-50</i>	0.8942	0.8263	0.8118	49.9820
<i>MODE-F-3o-CON-COR-MM-50</i>	0.8944	0.8302	0.8066	49.9640
<i>AllKNN-TSS</i>	0.8777	0.9854	0.8569	77.7751
<i>ENN-TSS</i>	0.8853	0.9754	0.8560	84.6832
<i>ENNTh-TSS</i>	0.8686	0.9891	0.8537	64.6310
<i>ENRBF-TSS</i>	0.8453	1.0000	0.8453	84.5255
<i>MENN-TSS</i>	0.8684	0.9891	0.8537	64.6202
<i>ModelCS-TSS</i>	0.9496	0.9050	0.8316	90.7993
<i>Multiedit-TSS</i>	0.8646	0.9903	0.8528	79.1944
<i>NCNedit-TSS</i>	0.8997	0.9610	0.8538	84.5886
<i>POP-TSS</i>	0.9695	0.7276	0.7874	67.3146
<i>PSRCG-TSS</i>	0.9221	0.4572	0.7569	36.5829
<i>RNG-TSS</i>	0.8938	0.9695	0.8586	85.9403
<i>VSM-TSSs0</i>	0.6997	0.4693	0.6094	24.7923

Table C.22

Win-loss stratified 10-fold cross-validation ranking test for *WINE RED*.

Method	$\overline{ACC}_D^{1-NN}$			$\overline{ACC}_{CV}^{1-NN}$			$\overline{ACC}_E^{1-NN}$			$\overline{RET}$		
	win	loss	dif.	win	loss	dif.	win	loss	dif.	win	loss	dif.
<i>NSGA-II-W-3o-1NN-ACC-R-0</i>	25	5	20	13	15	-2	22	1	21	4	27	-23
<i>MODE-W-3o-1NN-ACC-R-0</i>	27	1	26	13	14	-1	23	0	23	5	27	-22
<i>NSGA-II-F-3o-CON-COR-R-0</i>	24	5	19	13	12	1	22	3	19	3	27	-24
<i>MODE-F-3o-CON-COR-R-0</i>	25	3	22	14	13	1	22	0	22	3	29	-26
<i>NSGA-II-W-2o-1NN-ACC-R-0</i>	22	11	11	2	30	-28	19	0	19	11	19	-8
<i>MODE-W-2o-1NN-ACC-R-0</i>	24	5	19	10	22	-12	22	0	22	9	25	-16
<i>NSGA-II-W-3o-1NN-ACC-MM-0</i>	29	1	28	17	12	5	22	0	22	4	27	-23
<i>MODE-W-3o-1NN-ACC-MM-0</i>	28	1	27	15	12	3	24	0	24	3	29	-26
<i>NSGA-II-F-3o-CON-COR-MM-0</i>	26	1	25	14	13	1	25	0	25	1	33	-32
<i>MODE-F-3o-CON-COR-MM-0</i>	29	1	28	13	13	0	23	0	23	1	33	-32
<i>NSGA-II-W-2o-1NN-ACC-MM-0</i>	22	9	13	6	26	-20	18	0	18	11	18	-7
<i>MODE-W-2o-1NN-ACC-MM-0</i>	25	1	24	10	22	-12	22	0	22	9	25	-16
<i>NSGA-II-W-3o-1NN-ACC-R-50</i>	12	22	-10	5	26	-21	1	17	-16	24	5	19
<i>MODE-W-3o-1NN-ACC-R-50</i>	14	18	-4	10	22	-12	1	13	-12	19	11	8
<i>NSGA-II-F-3o-CON-COR-R-50</i>	10	24	-14	20	12	8	1	16	-15	19	11	8
<i>MODE-F-3o-CON-COR-R-50</i>	6	28	-22	24	9	15	1	16	-15	19	11	8
<i>NSGA-II-W-2o-1NN-ACC-R-50</i>	12	22	-10	0	35	-35	1	15	-14	32	1	31
<i>MODE-W-2o-1NN-ACC-R-50</i>	15	14	1	2	30	-28	5	11	-6	27	3	24
<i>NSGA-II-W-3o-1NN-ACC-MM-50</i>	17	15	2	5	26	-21	2	15	-13	24	7	17
<i>MODE-W-3o-1NN-ACC-MM-50</i>	14	19	-5	10	14	-4	4	12	-8	19	10	9
<i>NSGA-II-F-3o-CON-COR-MM-50</i>	5	27	-22	24	8	16	1	21	-20	19	10	9
<i>MODE-F-3o-CON-COR-MM-50</i>	7	26	-19	24	9	15	1	15	-14	19	11	8
<i>NSGA-II-W-2o-1NN-ACC-MM-50</i>	14	18	-4	2	30	-28	5	15	-10	30	2	28
<i>MODE-W-2o-1NN-ACC-MM-50</i>	17	14	3	2	28	-26	1	14	-13	25	4	21
<i>AllKNN-TSS</i>	4	31	-27	31	3	28	1	16	-15	29	3	26
<i>ENN-TSS</i>	5	29	-24	30	5	25	4	15	-11	18	17	1
<i>ENNTh-TSS</i>	1	32	-31	33	1	32	1	20	-19	33	1	32
<i>ENRBF-TSS</i>	0	35	-35	35	0	35	0	35	-35	25	7	18
<i>MENN-TSS</i>	1	32	-31	33	1	32	1	20	-19	35	0	35
<i>ModelCS-TSS</i>	17	15	2	26	8	18	17	4	13	11	22	-11
<i>Multiedit-TSS</i>	1	32	-31	31	3	28	1	21	-20	27	5	22
<i>NCNedit-TSS</i>	10	24	-14	28	6	22	5	16	-11	14	19	-5
<i>POP-TSS</i>	35	0	35	13	17	-4	23	0	23	0	35	-35
<i>PSRCG-TSS</i>	22	11	11	1	34	-33	18	11	7	12	21	-9
<i>RNG-TSS</i>	8	26	-18	28	6	22	10	13	-3	16	18	-2
<i>VSM-TSSs0</i>	19	14	5	6	26	-20	1	15	-14	13	20	-7

Table C.23

Win-loss stratified 10-fold cross-validation ranking test for *WINE WHITE*.

Method	$ACC_D^{1-NN}$			$ACC_{CV}^{1-NN}$			$ACC_E^{1-NN}$			$RET$		
	win	loss	dif.	win	loss	dif.	win	loss	dif.	win	loss	dif.
<i>NSGA-II-F-3o-CON-COR-R-0</i>	15	1	14	3	10	-7	15	0	15	1	15	-14
<i>MODE-F-3o-CON-COR-R-0</i>	15	1	14	3	10	-7	15	0	15	1	15	-14
<i>NSGA-II-F-3o-CON-COR-MM-0</i>	15	1	14	3	10	-7	15	0	15	1	15	-14
<i>MODE-F-3o-CON-COR-MM-0</i>	15	1	14	3	9	-6	15	0	15	1	15	-14
<i>NSGA-II-F-3o-CON-COR-R-50</i>	9	8	1	2	15	-13	5	8	-3	10	6	4
<i>MODE-F-3o-CON-COR-R-50</i>	7	11	-4	8	9	-1	5	8	-3	10	6	4
<i>NSGA-II-F-3o-CON-COR-MM-50</i>	7	10	-3	2	9	-7	3	8	-5	10	6	4
<i>MODE-F-3o-CON-COR-MM-50</i>	8	9	-1	2	10	-8	5	8	-3	10	6	4
<i>AllKNN-TSS</i>	4	15	-11	15	4	11	4	9	-5	17	2	15
<i>ENN-TSS</i>	5	14	-9	14	5	9	2	13	-11	14	5	9
<i>ENNTh-TSS</i>	2	16	-14	16	1	15	1	16	-15	18	1	17
<i>ENRBF-TSS</i>	0	19	-19	19	0	19	0	19	-19	15	4	11
<i>MENN-TSS</i>	1	17	-16	16	1	15	1	15	-14	19	0	19
<i>ModelCS-TSS</i>	12	7	5	11	8	3	12	6	6	5	13	-8
<i>Multiedit-TSS</i>	1	16	-15	16	1	15	1	14	-13	16	3	13
<i>NCNEdit-TSS</i>	10	8	2	12	7	5	5	8	-3	8	11	-3
<i>POP-TSS</i>	19	0	19	3	10	-7	15	0	15	0	19	-19
<i>PSRCG-TSS</i>	14	5	9	0	19	-19	14	5	9	5	13	-8
<i>RNG-TSS</i>	6	13	-7	13	6	7	6	8	-2	9	10	-1
<i>VSM-TSSs0</i>	13	6	7	1	18	-17	12	6	6	7	12	-5

Table C.24

Win-loss stratified 10-fold cross-validation ranking test for *SHOPPER*.

Method	$ACC_D^{1-NN}$			$ACC_{CV}^{1-NN}$			$ACC_E^{1-NN}$			$RET$		
	win	loss	dif.	win	loss	dif.	win	loss	dif.	win	loss	dif.
<i>NSGA-II-F-3o-CON-COR-R-0</i>	16	0	16	3	12	-9	3	9	-6	0	16	-16
<i>MODE-F-3o-CON-COR-R-0</i>	16	0	16	3	13	-10	3	9	-6	0	16	-16
<i>NSGA-II-F-3o-CON-COR-MM-0</i>	16	0	16	4	12	-8	5	9	-4	0	16	-16
<i>MODE-F-3o-CON-COR-MM-0</i>	16	0	16	3	12	-9	3	10	-7	0	16	-16
<i>NSGA-II-F-3o-CON-COR-R-50</i>	12	7	5	3	12	-9	3	9	-6	14	3	11
<i>MODE-F-3o-CON-COR-R-50</i>	6	12	-6	10	9	1	3	9	-6	16	2	14
<i>NSGA-II-F-3o-CON-COR-MM-50</i>	8	9	-1	8	10	-2	3	9	-6	14	3	11
<i>MODE-F-3o-CON-COR-MM-50</i>	8	9	-1	8	10	-2	3	10	-7	14	2	12
<i>AllKNN-TSS</i>	5	14	-9	15	4	11	16	0	16	10	9	1
<i>ENN-TSS</i>	6	12	-6	14	5	9	14	0	14	6	12	-6
<i>ENNTh-TSS</i>	4	15	-11	16	1	15	13	2	11	12	7	5
<i>ENRBF-TSS</i>	1	18	-17	19	0	19	12	7	5	7	11	-4
<i>MENN-TSS</i>	3	16	-13	16	1	15	13	2	11	13	6	7
<i>ModelCS-TSS</i>	14	5	9	11	8	3	11	8	3	4	15	-11
<i>Multiedit-TSS</i>	2	17	-15	16	1	15	13	3	10	9	10	-1
<i>NCNEdit-TSS</i>	11	8	3	12	7	5	13	0	13	6	11	-5
<i>POP-TSS</i>	15	4	11	2	17	-15	2	17	-15	11	8	3
<i>PSRCG-TSS</i>	13	6	7	0	19	-19	1	18	-17	18	1	17
<i>RNG-TSS</i>	8	9	-1	13	6	7	16	0	16	5	14	-9
<i>VSM-TSSs0</i>	0	19	-19	1	18	-17	0	19	-19	19	0	19



Table C.25

Performance of the 1 – NN, C4.5 and Random Forest classifiers using the selected instance set (solution x\*) for the WINE-QUALITY-RED dataset.

Method	$ACC_{D}^{1-NN}(x^*)$	$ACC_{CV}^{1-NN}(x^*, .5)$	$R_{D}^{1-NN}(x^*)$	$ACC_{T}^{C4.5}(x^*)$	$ACC_{D}^{C4.5}(x^*)$	$ACC_{CV}^{C4.5}(x^*, .5)$	$R_{D}^{C4.5}(x^*)$	$ACC_{T}^{RF}(x^*)$	$ACC_{D}^{RF}(x^*)$	$ACC_{CV}^{RF}(x^*, .5)$	$R_{D}^{RF}(x^*)$	$RET(x^*)$
NSGA-II-W-3o-1NN-ACC-R-0	0.9950	0.6331	0.0050	0.8987	0.8949	0.6089	0.0150	1.0000	0.9950	0.6949	0.0130	98.1864
MODE-W-3o-1NN-ACC-R-0	0.9981	0.6256	0.0117	0.9133	0.9112	0.5857	0.0203	1.0000	0.9975	0.6926	0.0153	98.8743
NSGA-II-W-3o-C4.5-ACC-R-0	0.9912	0.6264	0.0109	0.9338	0.9300	0.6199	<b>-0.0170</b>	1.0000	0.9931	0.6842	0.0237	97.2483
MODE-W-3o-C4.5-ACC-R-0	0.9881	0.6304	0.0119	0.9355	0.9218	0.6121	-0.0090	1.0000	0.9875	0.6936	0.0143	95.9350
NSGA-II-W-3o-RF-ACC-R-0	0.9981	0.6202	0.0171	0.9178	0.9162	0.6014	0.0046	1.0000	0.9987	0.6849	0.0231	99.6248
MODE-W-3o-RF-ACC-R-0	0.9975	0.6232	0.0141	0.9191	0.9174	0.5994	0.0066	1.0000	0.9981	0.6928	0.0152	99.7498
NSGA-II-W-2o-1NN-ACC-R-0	1.0000	0.5323	0.1050	0.8922	0.8674	0.5406	0.0654	1.0000	0.9844	0.6264	0.0815	82.3640
MODE-W-2o-1NN-ACC-R-0	1.0000	0.6164	0.0208	0.8913	0.8831	0.6061	0.0269	1.0000	0.9969	0.6863	0.0217	96.6854
NSGA-II-F-3o-CON-COR-R-0	0.9981	0.6357	0.0019	0.8826	0.8799	0.5827	0.0300	1.0000	0.9981	0.6926	0.0154	99.0619
MODE-F-3o-CON-COR-R-0	0.9994	0.6393	6.0E-4	0.9098	0.9099	0.6024	0.0036	1.0000	1.0000	0.685	0.0229	99.8749
NSGA-II-W-3o-1NN-ACC-MM-0	1.0000	0.6394	<b>-0.0011</b>	0.9100	0.9099	0.5997	0.0063	1.0000	1.0000	0.691	0.0169	99.3746
MODE-W-3o-1NN-ACC-MM-0	0.9925	0.6304	0.0075	0.8913	0.8874	0.6019	0.0225	1.0000	0.9937	0.7029	0.0063	96.6229
NSGA-II-W-3o-C4.5-ACC-MM-0	0.9756	0.6206	0.0244	0.9424	0.9293	0.6009	0.0051	1.0000	0.9812	0.6795	0.0284	92.3077
MODE-W-3o-C4.5-ACC-MM-0	0.9825	0.6394	0.0175	0.9359	0.9237	0.6086	-0.0082	1.0000	0.9894	0.6983	0.0106	95.5597
NSGA-II-W-3o-RF-ACC-MM-0	0.9994	0.6411	6.0E-4	0.9018	0.8999	0.6247	0.0100	0.9994	0.9987	0.694	0.0140	99.3121
MODE-W-3o-RF-ACC-MM-0	0.9950	0.6326	0.0050	0.9004	0.8937	0.6047	0.0163	1.0000	0.9944	0.6872	0.0208	98.5616
NSGA-II-W-2o-1NN-ACC-MM-0	0.9912	0.4625	0.1747	0.8562	0.8186	0.4858	0.1202	1.0000	0.9681	0.5521	0.1558	72.6079
MODE-W-2o-1NN-ACC-MM-0	1.0000	0.6111	0.0262	0.8999	0.8949	0.5756	0.0304	1.0000	0.9981	0.6990	0.0090	96.8105
NSGA-II-F-3o-CON-COR-MM-0	0.9969	0.6388	0.0031	0.9045	0.9024	0.6104	0.0075	1.0000	0.9987	0.6926	0.0153	98.8743
MODE-F-3o-CON-COR-MM-0	0.9975	0.6344	0.0029	0.9039	0.9006	0.6062	0.0094	1.0000	0.9969	0.7117	<b>0.0031</b>	99.5622
NSGA-II-W-3o-1NN-ACC-R-50	0.8218	0.4741	0.1782	0.7790	0.6811	0.5473	0.2289	1.0000	0.8049	0.6098	0.1951	41.0256
MODE-W-3o-1NN-ACC-R-50	0.8468	0.5381	0.1532	0.8553	0.6879	0.5266	0.2220	1.0000	0.8355	0.6218	0.1645	49.2808
NSGA-II-W-3o-C4.5-ACC-R-50	0.7505	0.5613	0.2495	0.9336	0.7817	0.5397	0.1282	1.0000	0.8011	0.6176	0.1989	43.3396
MODE-W-3o-C4.5-ACC-R-50	0.7642	0.5833	0.2358	0.9468	0.7886	0.5402	<b>0.1213</b>	1.0000	0.7967	0.6336	0.2033	43.5272
NSGA-II-W-3o-RF-ACC-R-50	0.803	0.566	0.1970	0.8820	0.7280	0.5368	0.1820	1.0000	0.8318	0.6421	0.1682	49.2808
MODE-W-3o-RF-ACC-R-50	0.7974	0.5682	0.2026	0.8936	0.7230	0.5657	0.1870	1.0000	0.8512	0.6295	0.1488	49.9687
NSGA-II-W-2o-1NN-ACC-R-50	0.7405	0.3768	0.2605	0.7899	0.5866	0.4674	0.3233	1.0000	0.6973	0.5181	0.3027	17.2608
MODE-W-2o-1NN-ACC-R-50	0.8462	0.4704	0.1669	0.8330	0.6617	0.5063	0.2483	1.0000	0.7824	0.5978	0.2176	34.8343
NSGA-II-F-3o-CON-COR-R-50	0.7617	0.6408	0.2383	0.9199	0.7067	0.6658	0.2033	1.0000	0.7817	0.6959	0.2183	49.9687
MODE-F-3o-CON-COR-R-50	0.7073	0.7695	0.2927	0.9597	0.6779	0.7859	0.2320	1.0000	0.7092	0.8375	0.2908	49.6560
NSGA-II-W-3o-1NN-ACC-MM-50	0.8099	0.4737	0.1901	0.8278	0.6710	0.5279	0.2389	1.0000	0.7980	0.5885	0.2020	39.2120
MODE-W-3o-1NN-ACC-MM-50	0.8512	0.5623	<b>0.1488</b>	0.8226	0.7111	0.5535	0.1989	1.0000	0.8374	0.6277	0.1626	49.7186
NSGA-II-W-3o-C4.5-ACC-MM-50	0.7292	0.5409	0.2708	0.9318	0.7624	0.5561	0.1476	1.0000	0.7892	0.6121	0.2108	41.2758
MODE-W-3o-C4.5-ACC-MM-50	0.7692	0.5453	0.2308	0.9221	0.7786	0.5269	0.1313	1.0000	0.803	0.5977	0.1970	44.1526
NSGA-II-W-3o-RF-ACC-MM-50	0.7824	0.5402	0.2176	0.8254	0.6798	0.5784	0.2301	1.0000	0.8099	0.6194	0.1901	45.8412
MODE-W-3o-RF-ACC-MM-50	0.8061	0.5439	0.1939	0.8885	0.7592	0.5138	0.1507	1.0000	0.8337	0.6404	<b>0.1463</b>	49.9062
NSGA-II-W-2o-1NN-ACC-MM-50	0.7805	0.4191	0.2195	0.8292	0.6366	0.4966	0.2733	1.0000	0.7498	0.5558	0.2502	27.4547
MODE-W-2o-1NN-ACC-MM-50	0.8912	0.4664	0.1709	0.8248	0.6967	0.5007	0.2133	1.0000	0.8380	0.5823	0.1620	47.4672
NSGA-II-F-3o-CON-COR-MM-50	0.7611	0.6583	0.2389	0.8911	0.6948	0.6483	0.2151	1.0000	0.7786	0.7322	0.2214	49.9687
MODE-F-3o-CON-COR-MM-50	0.7217	0.7127	0.2783	0.9197	0.6685	0.7064	0.2414	1.0000	0.7192	0.7980	0.2808	49.8437
AIKNN-TSS	0.6560	0.9272	0.3440	0.9482	0.6385	0.8252	0.2714	1.0000	0.6754	0.8948	0.3246	38.6492
ENN-TSS	0.6948	0.8982	0.3052	0.9621	0.6867	0.8083	0.2233	1.0000	0.7129	0.8639	0.2871	52.8455
ENINTh-TSS	0.6304	0.9570	0.3696	0.9731	0.6054	0.8548	0.3046	1.0000	0.6391	0.9140	0.3609	23.2645
ENRBF-TSS	0.4997	0.9857	0.5003	0.9974	0.5184	0.9804	0.3915	1.0000	0.5003	0.9857	0.4997	47.9675
MENN-TSS	0.6285	0.9299	0.3715	0.9757	0.6054	0.8679	0.3046	1.0000	0.6341	0.9326	0.3659	23.202
ModelCS-TSS	0.8693	0.7547	0.1307	0.9285	0.8136	0.6661	0.0963	1.0000	0.8787	0.7595	0.1213	76.9856
Multiedit-TSS	0.6291	0.9540	0.3709	0.9799	0.6366	0.8532	0.2733	1.0000	0.6629	0.9108	0.3371	43.4647
NCNedit-TSS	0.7705	0.8710	0.2295	0.9725	0.7686	0.7486	0.1413	1.0000	0.7930	0.8605	0.2070	65.9162
POP-TSS	1.0000	0.6148	<b>0.0225</b>	0.9051	0.9062	0.5971	<b>0.0089</b>	1.0000	1.0000	0.6850	<b>0.0229</b>	98.8743
PSRCG-TSS	0.9681	0.4548	0.1825	0.8543	0.7999	0.5008	0.1101	1.0000	0.9744	0.5620	0.1460	74.6717
RNG-TSS	0.7380	0.8972	0.2620	0.9465	0.7067	0.808	0.2033	1.0000	0.7280	0.8825	0.2720	59.5997
VSM-TSSs0	0.8837	0.5189	0.1184	0.9126	0.7936	0.4991	0.1163	1.0000	0.9368	0.5640	0.1440	69.4184

Table C.26

Performance values of the 1 – NN, C4.5 and Random Forest classifiers using the selected instance set (solution x\*) for the WINE-QUALITY-WHITE dataset.

Method	$ACC_{D}^{1-NN}(x^*)$	$ACC_{CV}^{1-NN}(x^*, .5)$	$R_{D}^{1-NN}(x^*)$	$ACC_{T}^{C4.5}(x^*)$	$ACC_{D}^{C4.5}(x^*)$	$ACC_{CV}^{C4.5}(x^*, .5)$	$R_{D}^{C4.5}(x^*)$	$ACC_{T}^{RF}(x^*)$	$ACC_{D}^{RF}(x^*)$	$ACC_{CV}^{RF}(x^*, .5)$	$R_{D}^{RF}(x^*)$	$RET(x^*)$
NSGA-II-F-3o-CON-COR-R-0	0.9888	0.6364	0.0112	0.9047	0.8932	0.5770	0.0104	1.0000	0.9898	0.6765	0.0107	96.8150
MODE-F-3o-CON-COR-R-0	0.9888	0.6364	0.0112	0.9047	0.8932	0.5770	0.0104	1.0000	0.9898	0.6765	0.0107	96.8150
NSGA-II-F-3o-CON-COR-MM-0	0.9973	0.6332	<b>0.0053</b>	0.8968	0.8942	0.5809	<b>0.0065</b>	1.0000	0.9980	0.6770	<b>0.0102</b>	98.7342
MODE-F-3o-CON-COR-MM-0	0.9973	0.6332	<b>0.0053</b>	0.8968	0.8942	0.5809	<b>0.0065</b>	1.0000	0.9980	0.6770	<b>0.0102</b>	98.7342
NSGA-II-F-3o-CON-COR-R-50	0.7762	0.5802	0.2238	0.9069	0.7168	0.5443	<b>0.1803</b>	1.0000	0.8028	0.6509	0.1972	50.0000
MODE-F-3o-CON-COR-R-50	0.7436	0.6608	0.2564	0.8841	0.6615	0.6460	0.2356	1.0000	0.7440	0.7276	0.2560	49.8367
NSGA-II-F-3o-CON-COR-MM-50	0.7811	0.5553	<b>0.2189</b>	0.8873	0.7095	0.5218	0.1876	1.0000	0.8142	0.6227	<b>0.1858</b>	50.0000
MODE-F-3o-CON-COR-MM-50	0.7611	0.6382	0.2389	0.8902	0.6809	0.5676	0.2162	1.0000	0.7738	0.6754	0.2262	50.0000
AIKNN-TSS	0.6429	0.9047	0.3571	0.9562	0.6047	0.7767	0.2924	1.0000	0.6519	0.8643	0.3481	36.8518
ENN-TSS	0.6670	0.8872	0.3330	0.9537	0.6564	0.7434	0.2407	1.0000	0.6964	0.8293	0.3036	49.4079
ENINTh-TSS	0.5931	0.9391	0.4069	0.9623	0.5659	0.8425	0.3312	1.0000	0.5988	0.9121	0.4012	21.1311
ENRBF-TSS	0.4488	1.0000	0.5512	1.0000	0.4488	1.0000	0.4483	1.0000	0.4488	1.0000	0.5512	44.8755
MENN-TSS	0.5927	0.9284	0.4073	0.9632	0.5649	0.8267	0.3322	1.0000	0.5988	0.8974	0.4012	21.0902
ModelCS-TSS	0.8789	0.7457	0.1211	0.9105	0.7932	0.6286	0.1039	1.0000	0.8889	0.7401	0.1111	76.9089
Multiedit-TSS	0.5827	0.9422	0.4173	0.9743	0.6084	0.8405	0.2887	1.0000	0.6229	0.9042	0.3771	38.1380
NCNedit-TSS	0.7881	0.8620	0.2119	0.9518	0.7617	0.7273	0.1354	1.0000	0.8005	0.8577	0.1995	67.3132
POP-TSS	1.0000	0.6302	<b>0.0082</b>	0.9036	0.9057	0.5607	<b>0.0267</b>	1.0000	1.0000	0.6662	<b>0.0210</b>	97.4479
PSRCG-TSS	0.9749	0.4992	0.1392	0.8675	0.7918	0.4882	0.1053	1.0000	0.9551	0.5847	0.1026	76.4394
RNG-TSS	0.7134	0.8690	0.2866	0.9623	0.6907	0.7557	0.2064	1.0000	0.7109	0.8560	0.2891	55.1654
VSM-TSSs0	0.9057	0.5248	0.1136	0.9016	0.806	0.5003	0.0911	1.0000	0.9336	0.5792	0.1080	73.2136

Table C.27

Performance values of the 1 – NN, C4.5 and Random Forest classifiers using the selected instance set (solution  $x^*$ ) for the ONLINE-SHOPPERS dataset.

Method	$ACC_D^{1-NN}(x^*)$	$ACC_{CV}^{1-NN}(x^*, 5)$	$R_D^{1-NN}(x^*)$	$ACC_T^{C4.5}(x^*)$	$ACC_D^{C4.5}(x^*)$	$ACC_{CV}^{C4.5}(x^*, 5)$	$R_D^{C4.5}(x^*)$	$ACC_T^{RF}(x^*)$	$ACC_D^{RF}(x^*)$	$ACC_{CV}^{RF}(x^*, 5)$	$R_D^{RF}(x^*)$	$RET(x^*)$
NSGA-II-F-3o-CON-COR-R-0	0.9980	0.8132	0.0020	0.9375	0.9370	0.8934	0.0049	1.0000	0.9990	0.9017	0.0010	99.043
MODE-F-3o-CON-COR-R-0	0.9980	0.8132	0.0020	0.9375	0.9370	0.8934	0.0049	1.0000	0.9990	0.9017	0.0010	99.043
NSGA-II-F-3o-CON-COR-MM-0	0.9981	0.8125	<b>0.0019</b>	0.9410	0.9407	0.8926	<b>0.0012</b>	0.9999	0.9989	0.9024	0.0011	98.8402
MODE-F-3o-CON-COR-MM-0	0.9981	0.8123	0.0022	0.9409	0.9407	0.8938	0.0012	1.0000	0.9993	0.9015	<b>7.0E-4</b>	98.8808
NSGA-II-F-3o-CON-COR-R-50	0.9041	0.8164	<b>0.0959</b>	0.9500	0.9166	0.9022	0.0253	1.0000	0.9470	0.9127	0.0530	50.0000
MODE-F-3o-CON-COR-R-50	0.8878	0.8455	0.1122	0.9680	0.9003	0.9436	0.0416	1.0000	0.9141	0.9470	0.0859	49.9189
NSGA-II-F-3o-CON-COR-MM-50	0.9031	0.8036	0.0969	0.9506	0.9210	0.8922	<b>0.0209</b>	0.9998	0.9505	0.9013	<b>0.0495</b>	49.7891
MODE-F-3o-CON-COR-MM-50	0.8930	0.8247	0.1070	0.9589	0.9050	0.9239	0.0369	1.0000	0.9276	0.9325	0.0724	49.9757
AIKNN-TSS	0.8786	0.9860	0.1214	0.9913	0.8938	0.9825	0.0482	1.0000	0.8931	0.9846	0.1069	77.8994
ENIN-TSS	0.8856	0.9763	0.1144	0.9864	0.8972	0.9710	0.0447	1.0000	0.9075	0.9765	0.0925	84.7526
ENINTh-TSS	0.8692	0.9895	0.1308	0.9934	0.8758	0.9902	0.0661	1.0000	0.8834	0.9922	0.1166	64.8094
ENRBF-TSS	0.8453	1.0000	0.1547	1.0000	0.8453	1.0000	0.0967	1.0000	0.8453	1.0000	0.1547	84.5255
MENN-TSS	0.8690	0.9899	0.1310	0.9935	0.8741	0.9895	0.0678	1.0000	0.8814	0.9924	0.1186	64.8013
ModelCS-TSS	0.9498	0.9096	0.0502	0.9592	0.9256	0.9271	0.0163	1.0000	0.9625	0.9345	0.0375	90.8354
Multiedit-TSS	0.8659	0.9913	0.1341	0.9932	0.8861	0.9841	0.0558	1.0000	0.8848	0.9870	0.1152	79.1403
NCNedit-TSS	0.9006	0.9640	0.0994	0.9849	0.9046	0.9669	0.0373	0.9999	0.9149	0.9722	0.0851	84.6634
POP-TSS	0.9959	0.7939	<b>0.0205</b>	0.9339	0.9400	0.8813	<b>0.0123</b>	1.0000	0.9998	0.8935	<b>0.0069</b>	90.3082
PSRCG-TSS	0.9333	0.4758	0.3387	0.8734	0.9190	0.7960	0.0976	1.0000	0.9655	0.7991	0.1014	37.6561
RNG-TSS	0.8948	0.9685	0.1052	0.9868	0.8946	0.9755	0.0474	1.0000	0.9030	0.9796	0.0970	85.9935
VSM-TSSs0	0.7112	0.4691	0.3453	0.8626	0.8457	0.7087	0.1849	1.0000	0.9079	0.7508	0.1496	24.4444

Table C.28

Performance values of 1 – NN, C4.5 and Random Forest classifiers using the selected instance set (solution  $x^*$ ) for the ALL-AGENTS dataset.

Method	$ACC_D^{1-NN}(x^*)$	$ACC_{CV}^{1-NN}(x^*, 5)$	$R_D^{1-NN}(x^*)$	$ACC_T^{C4.5}(x^*)$	$ACC_D^{C4.5}(x^*)$	$ACC_{CV}^{C4.5}(x^*, 5)$	$R_D^{C4.5}(x^*)$	$ACC_T^{RF}(x^*)$	$ACC_D^{RF}(x^*)$	$ACC_{CV}^{RF}(x^*, 5)$	$R_D^{RF}(x^*)$	$RET(x^*)$
NSGA-II-W-3o-1NN-ACC-R-0	0.9481	0.4688	0.0519	0.9219	0.8182	0.4531	0.1039	1.0000	0.9351	0.5313	0.0649	83.1169
NSGA-II-W-3o-C4.5-ACC-R-0	0.9610	0.4571	0.0390	0.9286	0.9351	0.5143	0.0052	1.0000	0.9610	0.5000	0.0714	90.9091
NSGA-II-W-3o-RF-ACC-R-0	0.9610	0.4648	0.0390	0.8310	0.8052	0.3944	0.1251	1.0000	0.9610	0.6056	0.0390	92.2078
NSGA-II-W-2o-1NN-ACC-R-0	0.7922	0.2667	0.2139	0.9333	0.5455	0.3333	0.3766	1.0000	0.6753	0.4000	0.3247	19.4805
NSGA-II-F-3o-CON-COR-R-0	0.9610	0.4730	0.0390	0.8514	0.8182	0.5135	0.1039	1.0000	0.9610	0.5676	0.0390	96.1039
MODE-W-3o-1NN-ACC-R-0	0.9740	0.4848	0.0260	0.9394	0.8961	0.5000	0.0260	1.0000	0.9351	0.5606	0.0649	85.7143
MODE-W-3o-C4.5-ACC-R-0	0.9221	0.4444	0.0779	0.9683	0.9351	0.5397	<b>-0.0166</b>	1.0000	0.9091	0.5079	0.0909	81.8182
MODE-W-3o-RF-ACC-R-0	0.9221	0.4348	0.0779	0.9420	0.8831	0.4058	0.1137	1.0000	0.9610	0.5072	0.0642	89.6104
MODE-W-2o-1NN-ACC-R-0	0.8831	0.2727	0.2078	0.8636	0.6104	0.5000	0.3117	1.0000	0.6494	0.3182	0.3506	28.5714
MODE-F-3o-CON-COR-R-0	0.9610	0.4795	0.0390	0.8493	0.8182	0.5068	0.1039	1.0000	0.9610	0.5479	0.0390	94.8052
NSGA-II-W-3o-1NN-ACC-MM-0	0.9610	0.4561	0.0390	0.9298	0.8312	0.4386	0.0909	1.0000	0.9091	0.5088	0.0909	74.0260
NSGA-II-W-3o-C4.5-ACC-MM-0	0.9221	0.4063	0.0779	0.9688	0.9481	0.4844	0.0351	1.0000	0.9221	0.5313	0.0779	83.1169
NSGA-II-W-3o-RF-ACC-MM-0	0.9481	0.4462	0.0519	0.9385	0.8961	0.4462	0.0733	1.0000	0.9610	0.4462	0.1253	84.4156
NSGA-II-W-2o-1NN-ACC-MM-0	0.8182	0.3000	0.1818	0.9000	0.6234	0.2000	0.3195	1.0000	0.6494	0.4000	0.3506	25.9740
NSGA-II-F-3o-CON-COR-MM-0	0.9610	0.4795	0.0390	0.8493	0.8182	0.5205	0.1039	1.0000	0.9740	0.5890	<b>0.0260</b>	94.8052
MODE-W-3o-1NN-ACC-MM-0	1.0000	0.4630	<b>0.0176</b>	0.8704	0.7662	0.3889	0.1558	1.0000	0.9091	0.4444	0.1270	70.1299
MODE-W-3o-C4.5-ACC-MM-0	0.7662	0.4583	0.2338	0.9792	0.9091	0.5208	0.0130	1.0000	0.8182	0.4375	0.1818	62.3377
MODE-W-3o-RF-ACC-MM-0	0.9221	0.5362	0.0779	0.9130	0.8701	0.4203	0.0992	1.0000	0.9481	0.5942	0.0519	89.6104
MODE-W-2o-1NN-ACC-MM-0	0.9481	0.2632	0.2174	0.8947	0.6494	0.3421	0.2727	1.0000	0.8182	0.2105	0.3609	49.3506
MODE-F-3o-CON-COR-MM-0	0.9740	0.4730	0.0260	0.9324	0.9091	0.5135	0.0130	1.0000	0.9740	0.5405	0.0309	96.1039
AIKNN-TSS	0.5584	0.9000	0.4416	1.0000	0.5065	0.7000	0.4156	1.0000	0.5065	0.9000	0.4935	25.9740
ENIN-TSS	0.5974	0.9063	0.4026	0.9688	0.6494	0.6875	0.2727	1.0000	0.6234	0.8438	0.3766	41.5584
ENINTh-TSS	0.5195	0.7000	0.4805	1.0000	0.5065	0.5000	0.4156	1.0000	0.4675	0.7000	0.5325	12.9870
ENRBF-TSS	0.6364	0.9211	0.3636	0.9737	0.6234	0.6579	0.2987	1.0000	0.6364	0.8684	0.3636	49.3506
MENN-TSS	0.4545	0.7500	0.5455	0.8750	0.3636	0.5000	0.5584	1.0000	0.4675	0.7500	0.5325	10.3896
ModelCS-TSS	0.7922	0.7255	0.2078	0.9608	0.7403	0.5490	0.1818	1.0000	0.8182	0.6078	0.1818	66.2338
Multiedit-TSS	0.4286	0.8636	0.5714	1.0000	0.4286	0.6818	0.4935	1.0000	0.4805	0.8636	0.5195	28.5714
NCNedit-TSS	0.6234	0.8529	0.3766	0.9412	0.5844	0.5882	0.3377	1.0000	0.6364	0.7941	0.3636	44.1558
POP-TSS	1.0000	0.4805	0.0000	0.9221	0.9221	0.5195	0.0000	1.0000	1.0000	0.5714	0.0000	100.0000
PSRCG-TSS	0.9870	0.3906	<b>0.0899</b>	0.9063	0.8701	0.3281	0.1914	1.0000	0.9481	0.4219	0.1496	83.1169
RNG-TSS	0.6234	0.8529	0.3766	0.9706	0.6364	0.5588	0.2857	1.0000	0.6364	0.7941	0.3636	44.1558
VSM-TSSs0	0.8961	0.4531	0.1039	0.9531	0.9091	0.3438	<b>0.1757</b>	1.0000	0.9610	0.5313	<b>0.0402</b>	83.1169

## Appendix D. Hypervolume confidence intervals

See Figs. D.8 and D.10.

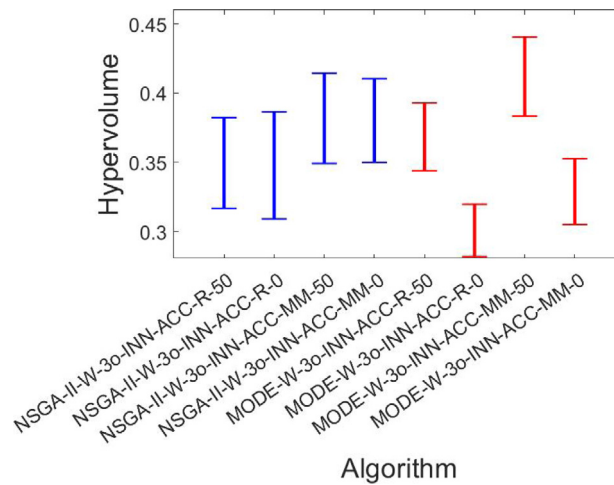


Fig. D.8. Hypervolume confidence intervals for WINE-QUALITY-RED dataset.

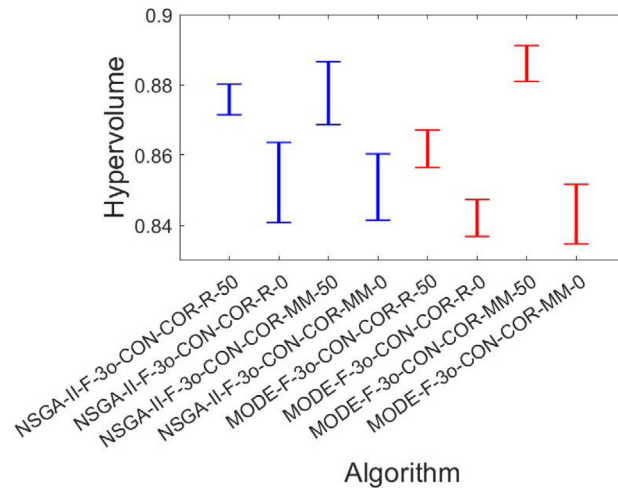


Fig. D.9. Hypervolume confidence intervals for WINE-QUALITY-WHITE dataset.

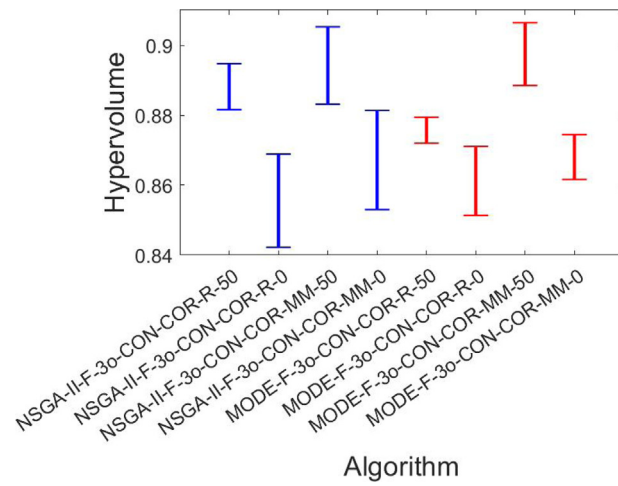


Fig. D.10. Hypervolume confidence intervals for ONLINE-SHOPPERS dataset.

## Appendix E. Pareto fronts of the best solutions in *ALL-AGENTS* datasets

See Figs. E.11–E.13.

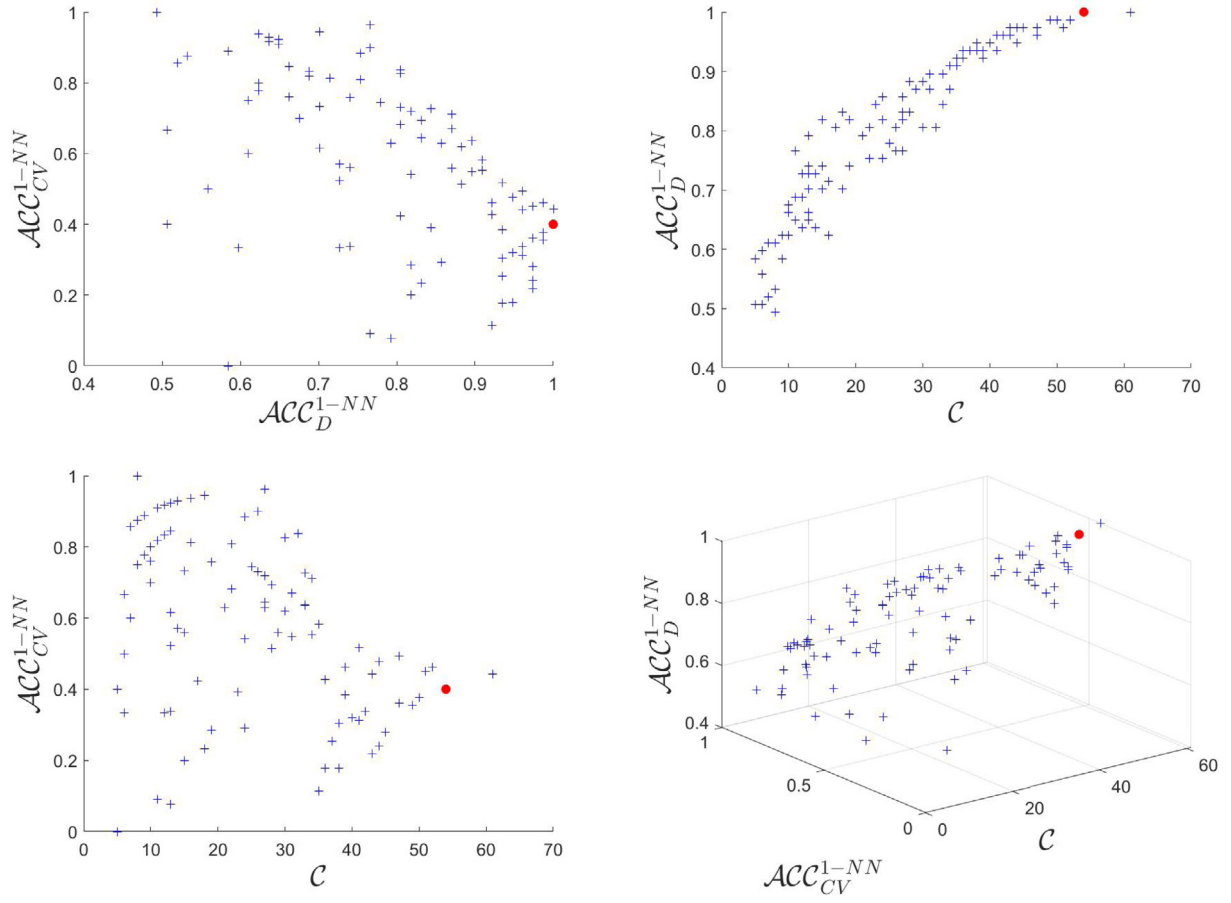


Fig. E.11. Non-dominated solutions in the last population and chosen solution (red circle) of *MODE-W-3σ-1NN-ACC-MM-0* for *ALL-AGENTS* dataset, in 2D and 3D.



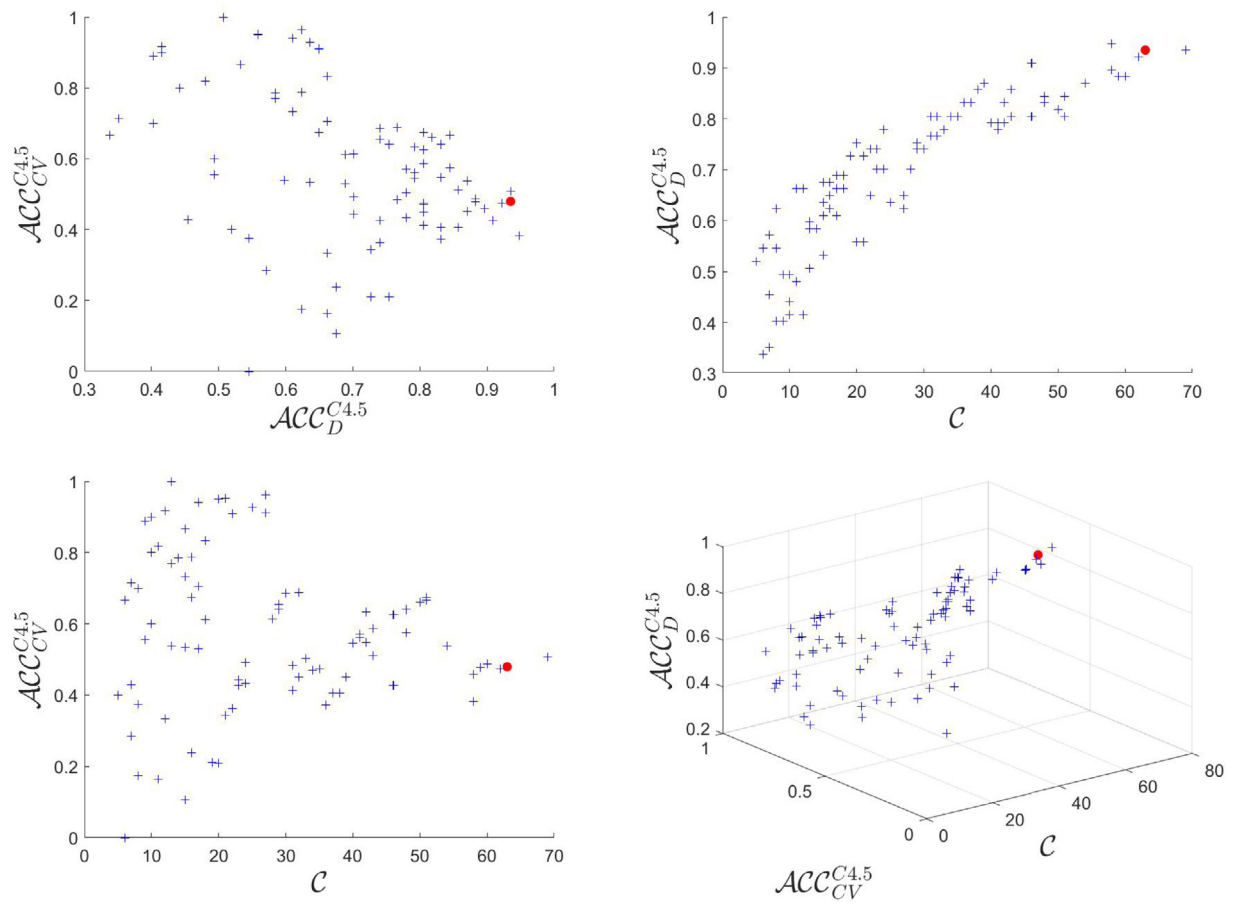


Fig. E.12. Non-dominated solutions in the last population and chosen solution (red circle) of *MODE-W-3o-C4.5-ACC-R-0* for *ALL-AGENTS* dataset, in 2D and 3D..

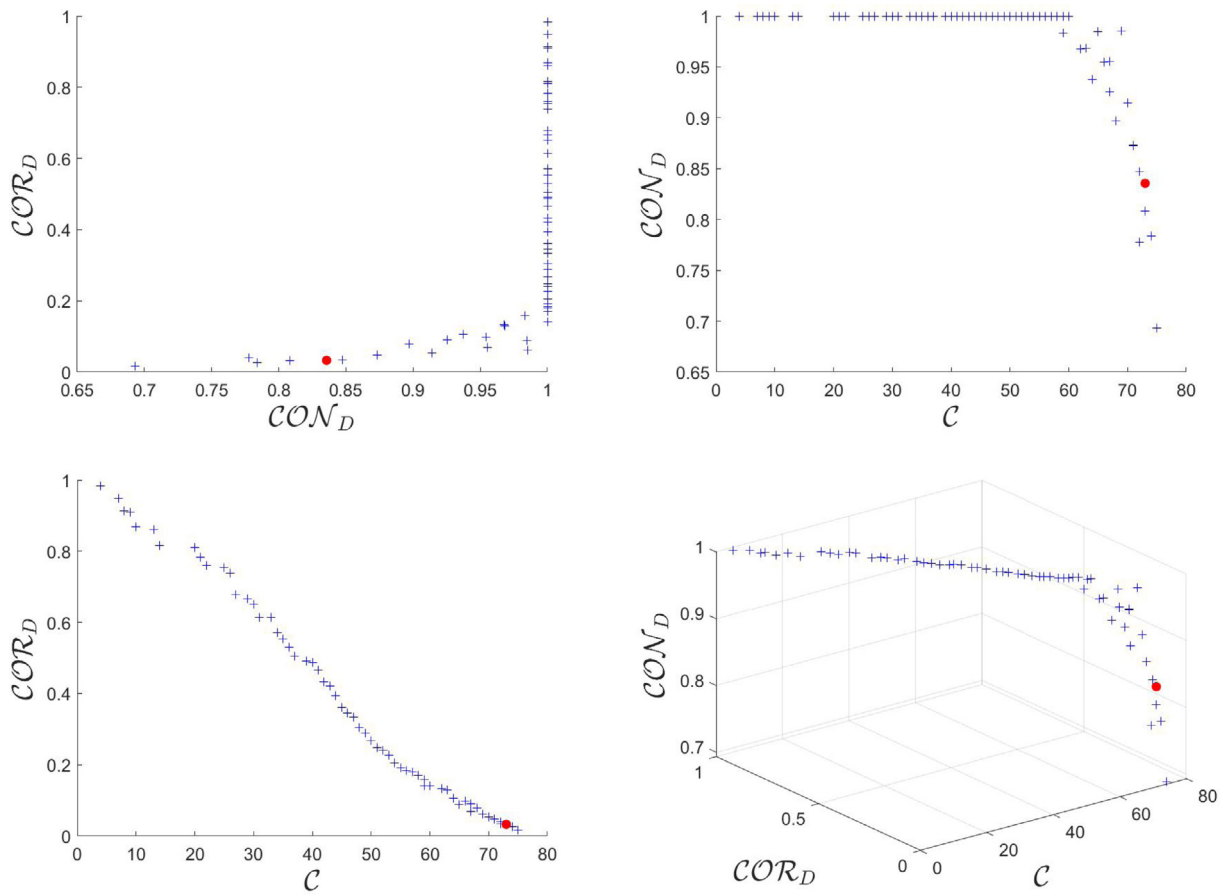


Fig. E.13. Non-dominated solutions in the last population and chosen solution (red circle) of NSGA-II-F-3o-CON-COR-MM-0 for ALL-AGENTS dataset, in 2D and 3D.

**Appendix F. Abbreviations**

ACC	Accuracy
AEI	Spanish Agency for Research
AllKNN	All k-Nearest Neighbors
AMPSO	Adaptive Michigan Particle Swarm Optimization
CFS	Correlation-based Feature Selection
CHC	Cross-generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation
CIE	Complementary Incremental Ensemble
CNN	Condensed Nearest Neighbor
DE	Differential Evolution
DE-C	Discretization-based Condensed Nearest Neighbor Ensemble
DE-E	Discretization-based Edited Nearest Neighbor Ensemble
DROP5	Decremental Reduction Optimization Procedure 5
EMOMIS-PbE	Evolutionary Multi-Objective Model and Instance Selection with Pareto-based Ensemble
ENN	Edited Nearest Neighbor
ENNNth	Edited Nearest Neighbor with Estimation of Probabilities Threshold
ENRBF	Edited Normalized Radial Basis Function
FEDER	European Fund for Regional Development
GENN	Generalized Editing using Nearest Neighbor
GPE	Global Pareto Ensemble
HVR	Hypervolume Ratio
IERE	Incremental Error Reduction Ensemble
IGA	Name of the evolutionary algorithm given in <a href="#">Ho et al. (2004)</a>
IMOEa	Name of the multi-objective evolutionary algorithm given in <a href="#">Ho et al. (2004)</a>
KDD	Knowledge Discovery in Databases
KEEL	Knowledge Extraction based on Evolutionary Learning
LVQTC	Learning Vector Quantization with Training Counter
MCIU	Spanish Ministry of Science, Innovation and Universities
MDE	Margin Distance based Ensemble
MENN	Modified Edited Nearest Neighbor
MOCHC	Name of the multi-objective evolutionary algorithm given in <a href="#">Rathee et al. (2019)</a>
MODE	Multi-Objective Differential Evolution
ModelCS	Model Class Selection
MOEA	Multi-Objective Evolutionary Algorithm
MSE	Name of prototyping method given in <a href="#">Triguero et al. (2012)</a> and <a href="#">Alcalá-Fdez et al. (2010)</a>
Multiedit	Multiple Edited Nearest Neighbor
NCNEdit	Nearest Centroid Neighborhood Edition
NN	Nearest Neighbors
NSGA-II	Non-dominated Sorting Genetic Algorithm II
NSGA-III	Non-dominated Sorting Genetic Algorithm III
PAES	Pareto Archived Evolution Strategy
PESA-II	Pareto Envelope based Selection II
POP	Pattern by Ordered Projections
PSCSA	Prototype Selection Clonal Selection Algorithm
PSO	Particle Swarm Optimization
PSRCG	Prototype Selection by Relative Certainty Gain
RET	Retention
RF	Random Forests
RNG	Prototype Selection based on Relative Neighborhood Graphs
SITSUS	Clinical Decision Support System for Infection Surveillance
SPEA	Strength Pareto Evolutionary Algorithm
SVM	Support Vector Machine
TE-C	Threshold-based Condensed Nearest Neighbor Ensemble
TE-E	Threshold-based Edited Nearest Neighbor Ensemble
t-SNE	t-distributed Stochastic Neighbor Embedding
TSS	Training Set Selection
UCI	University of California, Irvine
VSM	Variable Similarity Metric
Weka	Waikato Environment for Knowledge Analysis

## References

- Acampora, G., Herrera, F., Tortora, G., Vitiello, A., 2018. A multi-objective evolutionary approach to training set selection for support vector machine. *Knowl. Based Syst.* 147, 94–108.
- Aha, W., Kibler, D., K. Albert, M., 1991. Instance-based learning algorithms. *Mach. Learn.* 6, 37–66. <http://dx.doi.org/10.1023/A:1022689900470>.
- Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., Herrera, F., 2010. KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult.-Valued Logic Soft Comput.* 17, 255–287.
- Almuallim, H., Dietterich, T.G., 1991. Learning with many irrelevant features. In: AAAI. vol. 91, Citeseer, pp. 547–552.
- Arnaiz-González, A., Blachnik, M., Kordos, M., García-Osorio, C., 2016a. Fusion of instance selection methods in regression tasks. *Inf. Fusion* 30, 69–79.
- Arnaiz-González, A., Díez-Pastor, J.-F., Díez, J.J.R., García-Osorio, C.I., 2016b. Instance selection for regression: Adapting DROP. *Neurocomputing* 201, 66–81.
- Arnaiz-González, A., Díez-Pastor, J.F., Rodríguez, J.J., García-Osorio, C.I., 2016c. Instance selection for regression by discretization. *Expert Syst. Appl.* 54 (C), 340–350.
- Arnaiz-González, A., Díez-Pastor, J.-F., Rodríguez, J., García-Osorio, C., 2016d. Instance selection of linear complexity for big data. *Knowl.-Based Syst.* 107.
- Bertsekas, D., 1999. *Nonlinear Programming*, second ed. Athena Scientific, Cambridge, MA.
- Breiman, L., 2001. Random forests. *Mach. Learn.* 45 (1), 5–32.
- Brighton, H., Mellish, C., 2002. Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.* 6 (2), 153–172.
- Brodley, C., 1993. Addressing the selective superiority problem: Automatic algorithm/model class selection. In: 10th International Machine Learning Conference. ICML'93, pp. 17–24.
- Brunello, A., 2015. A Data Warehouse for a Contact Center with Multiple Channels and Skills Master's thesis). University of Udine.
- Brunello, A., Jiménez, F., Marzano, E., Montanari, A., Sánchez, G., Sciavicco, G., 2019. Multiobjective evolutionary feature selection and fuzzy classification of contact centre data. *Expert Syst. J. Knowl. Eng.* 36 (3).
- Brunello, A., Jiménez, F., Marzano, E., Palma, J., Sánchez, G., Sciavicco, G., 2018. Towards semi-automatic human performance evaluation: The case study of a contact center. *Intell. Data Anal.* 22 (4), 867–880.
- Cano, J., Herrera, F., Lozano, M., 2004. Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. *IEEE Trans. Evol. Comput.* 7 (6), 561–575.
- Cano, J.R., Herrera, F., Lozano, M., 2005. Stratification for scaling up evolutionary prototype selection. *Pattern Recognit. Lett.* 26 (7), 953–963.
- Chen, J.-H., Chen, H.-M., Ho, S.-Y., 2005. Design of nearest neighbor classifiers: multi-objective approach. *Internat. J. Approx. Reason.* 40 (1), 3–22, Data Mining and Granular Computing.
- Chen, Y., He, F., Li, H., Zhang, D., Wu, Y., 2020. A full migration BBO algorithm with enhanced population quality bounds for multimodal biomedical image registration. *Appl. Soft Comput.* 93, 106335.
- Cheng, F., Chen, J., Qiu, J., Zhang, L., 2020. A subregion division based multi-objective evolutionary algorithm for SVM training set selection. *Neurocomputing* 394, 70–83.
- Cheng, F., Chu, F., Zhang, L., 2021. A multi-objective evolutionary algorithm based on length reduction for large-scale instance selection. *Inf. Sci.* 576, 105–121.
- Coello, C., Veldhuizen, D., Lamont, G., 2002. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/Plenum publishers, New York, NY, USA.
- Collette, Y., Siarry, P., 2004. *Multiobjective Optimization: Principles and Case Studies*. Springer Berlin Heidelberg.
- Corne, D.W., Jerram, N.R., Knowles, J.D., Oates, M.J., J. M., 2001. PESA-II: Region-based selection in evolutionary multiobjective optimization. In: *Proceedings of the Genetic and Evolutionary Computation Conference. GECCO 2001*, Morgan Kaufmann Publishers, pp. 283–290.
- Cover, T., Hart, P., 2006. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.* 13 (1), 21–27.
- Cumming, G., Calin-Jageman, R., 2017. *Introduction to the New Statistics: Estimation, Open Science, and beyond*. Taylor & Francis.
- Dash, M., Liu, H., 2003. Consistency-based search in feature selection. *Artificial Intelligence* 151 (1–2), 155–176.
- Dean Bennette, W., 2011. *Instance Selection for Simplified Decision Trees Through the Generation and Selection of Instance Candidate Subsets* (Ph.D. thesis). Iowa State University, Ames, Iowa.
- Deb, K., 2001. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, London, UK.
- Deb, K., 2005. Multi-objective optimization. In: Burke, E.K., Kendall, G. (Eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer US, Boston, MA, pp. 273–316.
- Deb, K., Jain, H., 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* 18 (4), 577–601. <http://dx.doi.org/10.1109/TEVC.2013.2281535>.
- Deb, K., Pratab, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* 6 (2), 182–197.
- Devijver, P., 1986. On the editing rate of the MULTIEDIT algorithm. *Pattern Recognit. Lett.* 4 (1), 9–12.
- Dhiman, G., 2019. ESA: a hybrid bio-inspired metaheuristic optimization approach for engineering problems. *Eng. Comput.* 37, 323–353.
- Dhiman, G., Kumar, V., 2018. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. *Knowl.-Based Syst.* 159, 20–50.
- Du, K.-L., Swamy, M.N.S., 2014. *Neural networks and statistical learning*. Springer London, London, pp. 83–126.
- Dua, D., Graff, C., 2017. *UCI machine learning repository*. URL <http://archive.ics.uci.edu/ml>.
- Elkano, M., Galar, M., Sanz, J., Bustince, H., 2018. CHI-PG: A fast prototype generation algorithm for Big Data classification problems. *Neurocomputing* 287, 22–33.
- Escalante, H.J., Marín-Castro, M., Morales-Reyes, A., Graff, M., Rosales-Pérez, A., Montes-y Gómez, M., Reyes, C.A., González, J.A., 2017. MOPG: a multi-objective evolutionary algorithm for prototype generation. *Pattern Anal. Appl.* 20 (1), 33–47.
- Eshelman, L.J., 1991. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In: *Foundations of Genetic Algorithms*. Elsevier, pp. 265–283.
- Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., 1996. From data mining to knowledge discovery: An overview. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (Eds.), *Advances in Knowledge Discovery and Data Mining*. American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 1–34.
- Garain, U., 2008. Prototype reduction using an artificial immune model. *Pattern Anal. Appl.* 11 (3), 353–363.
- García-Osorio, C., de Haro-García, A., García-Pedrajas, N., 2010. Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts. *Artificial Intelligence* 174 (5), 410–441.
- Grochowski, M., Jankowski, N., 2004a. Comparison of instance selection algorithms I. Algorithms survey. In: *VII International Conference on Artificial Intelligence and Soft Computing*. ICAISC'04, In: *Lecture Notes on Computer Science*, vol. 3070, Springer-Verlag, Zakopane, pp. 598–603.
- Grochowski, M., Jankowski, N., 2004b. Comparison of instance selection algorithms II. Results and comments. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (Eds.), *Artificial Intelligence and Soft Computing - ICAISC 2004*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 580–585.
- Hall, M.A., 1999. *Correlation-based Feature Selection for Machine Learning*. Tech. Rep., University of Waikato Hamilton.
- Hall, M.A., 2000. Correlation-based feature selection for discrete and numeric class machine learning. In: *Proceedings of the Seventeenth International Conference on Machine Learning. ICML '00*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 359–366.
- Hamidzadeh, J., Kashefi, N., Moradi, M., 2020. Combined weighted multi-objective optimizer for instance reduction in two-class imbalanced data problem. *Eng. Appl. Artif. Intell.* 90, 103500.
- Haoran, L., He, F., Chen, Y., Luo, J., 2020. Multi-objective self-organizing optimization for constrained sparse array synthesis. *Swarm Evol. Comput.* 58, 100743.
- Hart, P., 2006. The condensed nearest neighbor rule (corresp.). *IEEE Trans. Inf. Theor.* 14 (3), 515–516.
- Hattori, K., Takahashi, M., 2000. A new edited k-nearest neighbor rule in the pattern classification problem. *Pattern Recognit.* 33, 521–528.
- Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* 97, 849–872.
- Ho, S.-Y., Shu, L.-S., Chen, J.-H., 2004. Intelligent evolutionary algorithms for large parameter optimization problems. *IEEE Trans. Evol. Comput.* 8, 522–541.
- Hou, N., He, F., Zhou, Y., Chen, Y., 2020. An efficient GPU-based parallel tabu search algorithm for hardware/software co-design. *Front. Comput. Sci.* 14, 145316.
- Hubertus, T., Klaus, M., Eberhard, T., 2004. *Optimization Theory*. Kluwer Academic, Dordrecht.
- Jankowski, N., Grochowski, M., 2004. Comparison of instances selection algorithms I. Algorithms survey. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (Eds.), *Artificial Intelligence and Soft Computing*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 598–603.
- Jiménez, F., Gómez-Skarmeta, A., Sánchez, G., 2002. How evolutionary multiobjective optimization can be used for goals and priorities based optimization. In: *I Congreso Español de Algoritmos Evolutivos Y Bioinspirados. AEB'02*, pp. 460–465.
- Jiménez, F., Verdegay, J.L., 2001. Computational intelligence in theory and practice. In: *Advances in Soft Computing*. Physica-Verlag HD, Heidelberg, pp. 167–182.
- Jiménez, F., Verdegay, J.L., Gómez-Skarmeta, A., 1999. Evolutionary techniques for constrained multiobjective optimization problems. In: *Proc. of the 1999 Genetic and Evolutionary Computation Conference (GECCO'99), Workshop Program*, pp. 115–116.
- Jones, D., Tamiz, M., 2010. *Practical Goal Programming*, first ed. Springer.
- Karhoff, H., 1991. *Linear Programming*. Birkhäuser Basel, Boston, MA.
- Kitchin, R., Lauriat, T., 2015. Small data in the era of big data. *GeoJournal* 80, 463–475. <http://dx.doi.org/10.1007/s10708-014-9601-7>.
- Knowles, J.D., Corne, D.W., 2000. Approximating the nondominated front using the Pareto archived evolution strategy. *Evol. Comput.* 8 (2), 149–172.
- Kononenko, I., 1994. Estimating attributes: analysis and extensions of RELIEF. In: *European Conference on Machine Learning*. Springer, pp. 171–182.
- Kordos, M., Arnaiz-González, A., García-Osorio, C., 2019. Evolutionary prototype selection for multi-output regression. *Neurocomputing* 358, 309–320.

- Kordos, M., Kapa, K., 2018. Multi-objective evolutionary instance selection for regression tasks. *Entropy* 20 (10), 1–34.
- Li, Y., Hu, Z., Cai, Y., Zhang, W., 2005. Support vector based prototype selection method for nearest neighbor rules. In: Wang, L., Chen, K., Ong, Y.S. (Eds.), *Advances in Natural Computation*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 528–535.
- Liu, H., Motoda, H., 1998. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA.
- Liu, H., Motoda, H., 2001. *Instance Selection and Construction for Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA.
- Liu, H., Setiono, R., et al., 1996. A probabilistic approach to feature selection—a filter solution. In: *ICML*. vol. 96, Citeseer, pp. 319–327.
- Lleó, V., Ferri, F.J., 2001. Another move toward the minimum consistent subset: A tabu search approach to the condensed nearest neighbor rule. *IEEE Trans. Syst. Man Cybern. B* 31, 408–413. <http://dx.doi.org/10.1109/3477.931531>.
- Lowe, D., 1995. Similarity metric learning for a variable-kernel classifier. *Neural Comput.* 7 (1), 72–85.
- Lumini, A., Nanni, L., 2006. A clustering method for automatic biometric template selection. *Pattern Recognit.* 3.
- Luo, J., He, F., Yong, J., 2020. An efficient and robust bat algorithm with fusion of opposition-based learning and whale optimization algorithm. *Intell. Data Anal.* 24 (3), 581–606.
- Van der Maaten, L., Hinton, G., 2008. Visualizing data using t-SNE. *J. Mach. Learn. Res.* 9 (11).
- Michalewicz, Z., 1996. *Genetic Algorithms + Data Structures = Evolution Programs*, third ed. Springer-Verlag, London, UK.
- Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. *Adv. Eng. Softw.* 95, 51–67.
- Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., 2005. Sequential search for decremental edition. In: Gallagher, M., Hogan, J.P., Maire, F. (Eds.), *Intelligent Data Engineering and Automated Learning. IDEAL 2005*, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 280–285.
- Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., 2008. Prototype selection via prototype relevance. In: Ruiz-Shulcloper, J., Kropatsch, W.G. (Eds.), *Progress in Pattern Recognition, Image Analysis and Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 153–160.
- Olvera-López, J.A., Carrasco-Ochoa, J.A., Martínez-Trinidad, J.F., Kittler, J., 2010. A review of instance selection methods. *Artif. Intell. Rev.* 34 (2), 133–143.
- Paredes, R., Vidal, E., 2000. Weighting prototypes - a new editing approach. In: *Pattern Recognition, International Conference on*. vol. 2, IEEE Computer Society, Los Alamitos, CA, USA, 25, 26, 27, 28.
- Pighetti, R., Pallez, D., Precioso, F., 2015. Improving SVM training sample selection using multi-objective evolutionary algorithm and LSH. In: *Computational Intelligence, 2015 IEEE Symposium Series on, Computational Intelligence and Data Mining*. Cape Town, South Africa, pp. 1383–1390.
- Quinlan, J.R., 2014. *C4. 5: Programs for Machine Learning*. Elsevier.
- Rathee, S., Ratnoo, S., Ahuja, J., 2019. Instance selection using multi-objective CHC evolutionary algorithm. In: Fong, S., Akashe, S., Mahalle, P.N. (Eds.), *Information and Communication Technology for Competitive Strategies*. Springer Singapore, Singapore, pp. 475–484.
- Riquelme, J.C., Aguilar-Ruiz, J.S., Toro, M., 2003a. Finding representative patterns with ordered projections. *Pattern Recognit.* 36 (4), 1009–1018.
- Riquelme, J., Aguilar-Ruiz, J., Toro, M., 2003b. Finding representative patterns with ordered projections. *Pattern Recognit.* 36, 1009–1018.
- Rosales-Pérez, A., Escalante, H.J., Coello Coello, C.A., González, J., Reyes-García, C.A., 2014a. An evolutionary multi-objective approach for prototype generation. In: *Proceedings of the 2014 IEEE Congress on Evolutionary Computation. CEC 2014*, 2014. pp. 1100–1107.
- Rosales-Pérez, A., García, S., Gonzalez, J.A., Coello Coello, C.A., Herrera, F., 2017. An evolutionary multiobjective model and instance selection for support vector machines with Pareto-based ensembles. *IEEE Trans. Evol. Comput.* 21 (6), 863–877.
- Rosales-Pérez, A., González, J.A., Coello-Coello, C.A., Reyes-García, C.A., Escalante, H.J., 2014b. Evolutionary multi-objective approach for prototype generation and feature selection. In: Bayro-Corrochano, E., Hancock, E. (Eds.), *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer International Publishing, Cham, pp. 424–431.
- Sánchez, J., Barandela, R., Márques, A., Alejo, R., Badenas, J., 2003. Analysis of new techniques to obtain quality training sets. *Pattern Recognit. Lett.* 24, 1015–1022.
- Sánchez, J., Pla, F., Ferri, F., 1997. Prototype selection for the nearest neighbor rule through proximity graphs. *Pattern Recognit. Lett.* 18, 507–513.
- Sebban, M., Nock, R., Lallich, S., 2002. Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problems. *J. Mach. Learn. Res.* 3, 863–885.
- Sinha, S., 2006. *Mathematical Programming: Theory and Methods*. Elsevier Science Limited.
- Skalak, D.B., 1997. *Prototype Selection for Composite Nearest Neighbor Classifiers* (Ph.D. thesis). University of Massachusetts, Amherst, MA, USA, UMI Order No. GAX97-37585.
- Storn, R., Price, K., 1995. *Differential Evolution: A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. TR., International Computer Science Institute, ICSI.
- Tian, Y., Si, L., Zhang, X., Cheng, R., He, C., Tan, K., Jin, Y., 2021. Evolutionary large-scale multi-objective optimization: A survey. *ACM Comput. Surv.*
- Tomek, I., 1976. An experiment with the edited nearest-neighbor rule. *IEEE Trans. Syst. Man Cybern.* 6 (6), 448–452.
- Triguero, I., Derrac, J., García, S., Herrera, F., 2012. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Trans. Syst. Man Cybern. C Appl. Rev.* 42, 86–100.
- Vazquez, F., Sánchez, J., Pla, F., 2005. A stochastic approach to Wilson's editing algorithm. In: *2nd Iberian Conference on Pattern Recognition and Image Analysis. IbPRIA05*, In: *Lecture Notes on Computer Science*, vol. 3523, Springer, Estoril, pp. 35–42.
- Wagdy, A., 2016. A new modified binary differential evolution algorithm and its applications. *Appl. Math. Inf. Sci.* 10, 1965–1969.
- Wilson, D.L., 1972. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Trans. Syst. Man Cybern.* 2 (3), 408–421.
- Wilson, D.R., Martínez, T.R., 2000. Reduction techniques for instance-based learning algorithms. *Mach. Learn.* 38 (3), 257–286.
- Xue, F., Sanderson, A., Graves, R., 2004. Pareto-based multi-objective differential evolution. In: *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*. vol. 2, pp. 862–869.
- Yang, L., Zhu, Q., Huang, J., Wu, Q., Cheng, D., Hong, X., 2019. Constraint nearest neighbor for instance reduction. *Soft Comput.* 23 (24), 13235–13245.
- Zeleny, M., 1973. *Compromise programming*. In: Cochrane, J., Zeleny, M. (Eds.), *Multiple Criteria Decision Making*. University of South Carolina Press, Columbia, pp. 262–301.
- Zitzler, E., Thiele, L., 2000. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* 3, 257–271.
- Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., Grunert da Fonseca, V., 2002. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Trans. Evol. Comput.* 7, 117–132.