Soft Computing (2023) 27:12863–12881
https://doi.org/10.1007/s00500-022-06768-8

FOCUS

A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers

Rosita Guido[1] · Maria Carmela Groccia[1] · Domenico Conforti[1]

Accepted: 10 January 2022 / Published online: 2 February 2022

Abstract
In machine learning, hyperparameter tuning is strongly useful to improve model performance. In our research, we concentrate our attention on classifying imbalanced data by cost-sensitive support vector machines. We propose a multi-objective approach that optimizes model's hyper-parameters. The approach is devised for imbalanced data. Three SVM model's performance measures are optimized. We present the algorithm in a basic version based on genetic algorithms, and as an improved version based on genetic algorithms combined with decision trees. We tested the basic and the improved approach on benchmark datasets either as serial and parallel version. The improved version strongly reduces the computational time needed for finding optimized hyper-parameters. The results empirically show that suitable evaluation measures should be used in assessing the classification performance of classification models with imbalanced data.

Keywords Multi-objective optimization · Support vector machine · Hyper-parameter optimization · Imbalanced datasets · Genetic algorithms

1 Introduction

Classification problems may be encountered in different domains. One of these is the disease diagnosis, which establishes the presence or absence of a given disease according to referred symptoms and results of medical exams. Machine learning approaches can be employed to support experts in diseases diagnosis. Many researches aim to propose new methods to improve or enhance the outcomes of existing ones.

Support vector machines (SVM) are one of the best machine learning (ML) models for solving several real-life classification problems (Vapnik 1998; Cristianini and Shawe-Taylor 2000). The choice of hyper-parameters of a ML model can significantly affect the resulting model's performance. Generally, hyper-parameters are adjusted for each model in order to find a hyper-parameter setting that maximizes the model performances and so that the ML model can predict unknown data accurately. The goal of hyper-parameter optimization is to find a set of values that minimizes a predefined loss function.

Usually, a good set of hyper-parameters are determined by a grid search. The grid search strategy is based on testing all hyper-parameter combinations specified in a multi-dimensional grid. During the search, the hyper-parameters are varied, with fixed step-size, in a given range of values. The performance of a combination of hyperparameters is evaluated using a performance metric. The configuration with the best performance is selected and used to train the ML model on the whole dataset. However, this kind of search is very time consuming and it is suitable for the adjustment of few hyper-parameters.

Another big challenge in data mining that is attracting increasing interest of researchers is dealing with imbalanced data sets (Japkowicz and Stephen 2002). A dataset is imbalanced when one or more classes have very low proportions in the data as compared to the other classes. The first class is called as minority class with respect to the majority class(ess). The main interest is in correctly classifying

Communicated by Dario Pacciarelli.

✉ Rosita Guido
rosita.guido@unical.it

Maria Carmela Groccia
mariacarmela.groccia@unical.it

Domenico Conforti
domenico.conforti@unical.it

[1] Department of Mechanical, Energy and Management Engineering, University of Calabria, Ponte Pietro Bucci, 87036 Rende, Cosenza, Italy

the minority class. The existing methods for classification of imbalanced data can be categorized as algorithm-level category, data-level category, and cost-sensitive methods that lie between the above two categories (Galar et al. 2012). The first category includes methods modified or designed to handle imbalanced data; the second category includes methods that try to transform data in order to balance classes and use then standard classification algorithms. Down-sampling approaches, which reduce the majority class in the training subset, and over-sampling approaches, which increase the size of the minority class in the training subset, belong to this category. Finally, the third category includes methods designed for weighting differently the classes by introducing misclassification costs.

It is important to point out that the most commonly used model evaluation metric is the accuracy. However, it can be very misleading when data are imbalanced. In such cases, different evaluation metrics should be considered. We tested in (Guido et al. 2021) two evaluation model metrics, i.e., accuracy and G-Mean, on two imbalanced benchmark datasets by optimizing hyper-parameters of support vector machines by genetic algorithms (GAs). Comparing the results, we observed empirically that G-Mean is more suitable than accuracy to evaluate model performance in case of imbalanced data, especially when data refers to medical domains, like diagnosis. The results encouraged us to continue exploring this research field.

This research paper addresses the optimal hyper-parameters problem as a multi-objective problem. It has a twofold contribution:

1. The main goal is to investigate methods for improving hyper-parameter tuning of SVM. We propose a novel approach for optimal hyper-parameter tuning that consists of a genetic algorithm combined with a decision tree. The basic idea is that some chromosomes are similar among them and they have thus the same fitness value. A decision tree (DT), trained in a suitable manner, is exploited to reduce the number of k-fold cross-validation to be performed and thus the overall computational time. As we will see, GAs were chosen even because they allow for an easy parallelization of the problem, which is tremendously helpful. The approach that combines GAs and DT strongly reduces the overall computational time, as described in Sect. 5.
2. It focuses on testing and optimizing, at the same time, more suitable performance measures in addition to the accuracy. This is important for application domains where one data class is of more interest than others.

The paper is structured as follows. A short review of the state-of-the-art of the literature focusing on imbalanced data sets is in Sect. 2. We give a short description of support vector machines and decision trees in Sect. 3, and discuss some metrics commonly used to evaluate model performance. In Sect. 4, we introduce multi-objective optimization problems and the Non-dominated Sorting Genetic Algorithm-II (NSGA-II). In Sect. 5, we detail our approach that combines genetic algorithms and a heuristic procedure based on decision tree in order to find optimal hyper-parameters. Three objective functions are optimized. We perform several computational experiments aimed at finding the best hyper-parameter tuning for six benchmark datasets. The best results along with a discussion and comparison with other results of the literature are reported in Sect. 6. Finally, the conclusions are given in Sect. 7.

2 Related work on imbalanced data classification and cost-sensitive learning problems

Let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$, be a dataset where $x_i \in \Re^L$ is a pattern (even called example) drawn from a domain X and $y_i \in Y$ is its related class label. An example is thus a vector. In a binary classification domain, an example can be either positive, denoted by a label $y = 1$, or negative, denoted by $y = -1$. Generally, the goal of a binary classifier is to map feature vectors $x \in X$ to class labels $y \in \{\pm 1\}$. In terms of functions, a classifier can be written as $h(x) = sign[p(x)]$, where the function $p : X \rightarrow R$ is denoted as the classifier predictor.

Classifiers generally perform poorly on imbalanced datasets and, as a consequence, often they classify almost all instances as negative. In recent years, imbalanced data classification has been studied by many researchers with different methods (Jo and Japkowicz 2004; Galar et al. 2012). These methods can be distinguished into two categories based on data and algorithms. Data-based methods focus on data pre-processing to reduce imbalanced data. For instance, up-sampling and under-sampling are two methods that modify instance distribution. Up-sampling methods increase the minority samples, whereas under-sampling methods reduce the majority samples. Synthetic Minority Oversampling Technique is an oversampling method that balances data by generating new samples similar to the minority samples and their neighbors (Chawla et al. 2002).

Hereafter, a positive instance belongs to the minority class, whereas a negative instance to the majority class. In many real-world applications, misclassifications may have different costs, such as for instance disease diagnosis and business decision making. The related classification problem, called *cost-sensitive learning* problem, aims at minimizing the total misclassification costs. The issue of classifying imbalanced data by an SVM was addressed in (Veropoulos et al. 1999) by a biased-SVM. This method uses two penalty coefficients for

misclassified positive instances and negative instances. Since the positive instances usually belong to the minority class, the used penalty coefficient for this class is bigger than the penalty coefficient associated with the majority class. In this way, the SVM classifier aims at reducing misclassification rate of the minority class.

The performance of an SVM model even depends, for instance, on the used kernel function, which maps instances-vectors from the original input space to higher dimensional spaces to deal with nonlinearly separable data (Scholkopf and Smola 2001). Accordingly, two parameters of SVM, i.e., C and the kernel parameter were found by an exhaustive search approach in (Mehrbakhsh et al. 2019). Iranmehr et al. (2019) extended the SVM with cost-sensitive learning considering example dependent costs. They performed experimental analysis on class imbalance, cost-sensitive learning with a given class and example costs and showed that their proposed algorithm provides superior generalization performance compared to conventional methods. Qi et al. (2013) proposed a new Cost-Sensitive Laplacian SVM and tested its effectiveness via experiments on public datasets. They evaluate the algorithms performance by the Average Cost. Tao et al. (2019) developed a novel self-adaptive cost weights-based SVM cost-sensitive ensemble for imbalanced datasets classification tasks. The approach was tested on synthetic datasets and on public datasets showing higher classification accuracy than the other existing imbalanced classification methods in terms of G-Mean and F-Measure.

Evolutionary algorithms are flexible and commonly used for a plethora of machine learning problems and tasks (Bergstra et al. 2011; Goldberg and Holland 1988). Evolutionary optimization-based techniques solve the filter design task as an optimization problem. They are used successfully in different real-world optimization problems related to Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) digital filters design. The goal is to minimize an error function that quantifies deviation between a filter and a desired response. This error is reduced by updating iteratively a set of filter coefficients such that given specifications are met. Dwivedi et al. (2018) provided a comprehensive review of the various evolutionary optimization-based techniques for FIR filter design. Approaches to design IIR filters based on evolutionary techniques were proposed in (Agrawal et al. 2018, 2017). Evolutionary algorithms are even used to automatically tune several parameters. Lessmann et al. (2005) used a GA in order to tune SVMs. Phienthrakul and Kijsirikul (2010) improved the accuracy of SVM by a nonlinear combination of multiple RBF kernels to obtain more flexible kernel functions. The hyperparameters are chosen by an evolutionary strategy where the objective functions are based on training accuracy, bounding of generalization error, and subset cross-validation on training accuracy. The resulting kernel allows better discrimination in the feature space than that of a single RBF kernel.

One of the first research papers on cost-sensitive approach tackled with an evolutionary process is due to Turney (1995). Recently, Noia et al. (2020) applied SVM, k-Nearest Neighbors and k-means as clustering techniques to predict the probability of contracting a given disease starting from both workplace-related (using Ateco and Istat codes) and worker-related characteristics (i.e., age at hiring, age at disease certification, gender, employment duration). They used a GA to find the best values of the used methods. Misclassification error rate is used as fitness function. However, since the classes were not evenly distributed among the instances, they used a second fitness function that reduces the misclassification error rate of the minority class.

An exhaustive search of papers addressing evaluation of ML algorithms on classification is due to Sokolova et al. (2006). These authors showed that the clear "leaders" are those papers in which evaluation is performed on data from the UCI repository, in biomedical and medical sciences, visual and text classification, and language applications. The most used evaluation measures are accuracy, precision, recall, F-score, and the Receiver Operating Characteristic (ROC).

3 Learning model classifiers

We optimize hyper-parameters of SVM classifiers with Gaussian kernel in order to correctly compare our results found on public and well-known datasets with those reported in the literature. Our approach, as it is better detailed in Sect. 5, trains and uses random trees to reduce the overall computational time.

In this section, we briefly introduce SVM and decision trees. Thus, we report the most used performance metrics of a ML model and discuss their suitability in case of imbalanced datasets.

3.1 Support vector machine

The SVM was introduced by Cortes and Vapnik (1995) and is based on statistical learning theory (Vapnik 1998). SVMs are a class of algorithms for classification, regression and other applications (Cristianini and Shawe-Taylor 2000) and they are among the most used ML techniques.

Let X, be a dataset with L instances $X = (x_1, \ldots, x_L)$, where $x_i \in \Re^m$, denotes an instance with m features, and $y_i \in \{\pm 1\}$ its label, $i = 1, \ldots, L$. In a binary classification problem, an SVM basically searches for an optimal hyperplane that separates patterns of the two classes by maximizing the *margin* $w \in \Re^m$. Finding the optimal hyperplane means solving the quadratic programming model (1)-(3), which is known as soft-margin SVM

$$min\frac{1}{2}||w||^2 + C\sum_{1}^{L}\xi_i \tag{1}$$

$$y_i(w^T\phi(x_i) + b) - 1 + \xi_i \geq 0 \quad i = 1, \ldots, L \tag{2}$$

$$\xi_i \geq 0 \quad i = 1, \ldots, L \tag{3}$$

where C, named penalty parameter, is a trade-off between the size of the margin of separation w and the training errors ξ; b is the bias and it indicates the offset of the hyperplane from the origin. Constraints (2) state that when a training example x_i lies on the wrong side of the hyperplane, the corresponding slack variable ξ_i is greater than 1. Small values of C increase the training errors, whereas larger values bring it closer to the hard-margin SVM. In case of nonlinearly separable datasets, the SVM basically maps input vectors into high-dimensional feature spaces by the so-called kernel functions (Hofmann et al. 2008). A kernel function, denoted as $K(x_i, x_j) = \langle\phi(x_i), \phi(x_j)\rangle$, is an inner product in a feature space where it measures similarity between any pair of inputs x_i and x_j. A kernel function can take many different forms (Hofmann et al. 2008), such as

- Linear kernel $K(x_i, x_j) = (x_i^T x_j)^d$
- Polynomial kernel $K(x_i, x_j) = (x_i^T x_j + a)^d$
- Radial Basis Function (RBF) kernel $K(x_i, x_j) = exp(-\gamma\|x_i - x_j\|^2)$

The decision function, i.e., the classifier, is specified by a subset of training instances, the so-called *support vectors*, that are the only vectors that "support" the optimal separating hyperplane.

It is well known that the performance of most machine learning algorithms on a given dataset depends on well-tuned hyper-parameter. In setting up an SVM model, for instance, two problems are encountered: (1) how to select the kernel function, and (2) how to select its hyper-parameter. An SVM with polynomial kernel has three parameters that need to be optimized: the regularization parameter C, the parameter a, and the degree d. The optimization of these three parameters if 50 steps should be performed, requires an amount of time to test the total $50^3 = 125000$ combinations. The greater the number of parameters to be set, the greater is the number of combinations.

The cost-sensitive SVM (CS-SVM) uses two penalty weights for the two classes. Let C_1, be the cost of a false negative. It penalizes misclassification of instances of the minority class. Analogously, let C_{-1}, be the cost of a false positive. It penalizes misclassification of instances of the majority class. The optimization model CS-SVM is (4)-(6).

$$\underset{w,b,\xi}{\text{argmin}} \quad \frac{1}{2}\|w\|^2 + C[C_1\sum_{i|y_i=1}]\xi_i + C_{-1}\sum_{i|y_i=-1}\xi_i] \tag{4}$$

$$y_i(w^T x + b) \geq 1 - \xi_i \quad i = 1, \ldots, L \tag{5}$$

$$\xi_i \geq 0 \quad i = 1, \ldots, L \tag{6}$$

Observe that the cost matrices has the diagonal elements as zero— because of the assumption that a correct classification has no cost—and the off-diagonal elements are positive numbers. However, the range of possibilities for CS-SVM hyper-parameter can be huge.

Datta and Das (2015) proposes a Near-Bayesian Support Vector Machine (NBSVM) for imbalanced classification problems by combining decision boundary shift and unequal regularization costs. Extensive comparison with standard SVM and some state-of-the-art methods is furnished as a proof of the ability of the NBSVM to perform competitively on imbalanced datasets.

3.2 Decision tree

A decision tree is a supervised learning algorithm for regression and classification problems (Breiman et al. 1984) and is the most popular form of rule-based classifiers (Witten and Frank 2005). It has a set of elements called *nodes* and is built top-down from a root node. Each node represents a single input attribute: leaf nodes contain an output attribute, which is used to make a prediction; the other nodes are split points of an attribute. The data is partitioned into homogeneous subsets, i.e., they contain instances with similar values. Given a new input, the tree is traversed by evaluating the specific input started at the root node of the tree.

3.3 Performance evaluation and some limitations

To estimate the generalization performance of an SVM model, generally one evaluates accuracy measure on data not used for training the model. The k-fold cross-validation (k-CV) is the most used procedure. It consists on partitioning data into k disjoint sets of approximately equal size. An SVM is thus trained k times: at the $i-th$ iteration, all the disjoint sets are used as training set except the $i-th$ set, which is used to evaluate the performance of the model. The errors observed in this process are averaged yielding the k-fold CV error.

Before introducing the most used evaluation measures, it is useful to revise the confusion matrix of binary classification problems. A general confusion matrix is illustrated in Table 1. The two columns refer to the predicted classes, whereas the two rows refer to the actual classes. True Positives (TP) is the number of positive instances correctly classified and False Negatives (FN) is the number of positive instances incorrectly classified as negative. These two numbers refer to the minority class. Similarly, True Negatives (TN) is the number of negative instances correctly classified, and False Positives

Table 1 Confusion matrix for a binary problem

		Predicted	
		positive class	negative class
Actual	positive class	TP	FN
	negative class	FP	TN

(FP) is the number of negative instances incorrectly classified as positive class. These two numbers refer to the majority class. Observe that, in case of data related to patients, a false negative means that patient has the disease but the diagnosis result says that it does not have.

The most common evaluation measures used are listed below.

Accuracy defined as the ratio between the number of instances correctly classified and the total number of instances. It assesses the overall effectiveness of the model by showing the probability of the true value of the class label

$$Accuracy = \frac{TP + TN}{TP + FP + Tn + FN}$$

Other two measures that separately estimate a classifier's performance on different classes are sensitivity and specificity. They are often employed in medical and bio-medical applications.

Sensitivity (true positive rate) is defined as the ratio between the number of positive instances correctly classified as such and the number of positive instances

$$Sensitivity = \frac{TP}{TP + FN}$$

Specificity (true negative rate) is defined as the ratio between the number of negative instances correctly classified as such and the number of negative instances

$$Specificity = \frac{TN}{TN + FP}$$

Precision is defined as the ratio of TP to the number of all instances predicted as positive

$$Precision = \frac{TP}{TP + FP}$$

As reported especially recently in some papers (e.g., Tao et al. 2019), the accuracy-based evaluation measure is not suitable for classification of imbalanced data as the minority class has very little effect on the accuracy compared to the majority class. For imbalanced classification problems, the correct classification of instances of the minority class is usually the most important measure. There are further interesting classification evaluation measures that allow to balance false positive rate and false negative rate. Here, among these measures, we evaluate even F-Measure, the Geometric Mean, the average cost, the Youden's index, and the balanced accuracy. They are defined as follows.

F-Measure integrates sensitivity and precision into an average by a harmonic mean

$$F - Measure = \frac{2Sensitivity \times Precision}{Sensitivity + Precision}$$

The harmonic mean of two numbers tends to be closer to the smaller number. A high F-Measure value means that both Sensitivity and Precision are high.

Geometric Mean(G-Mean) is suggested as the balanced performance between the two classes. It is intrinsically defined as the geometric mean of sensitivity and specificity. If the G-Mean value is high, both Sensitivity and Specificity are expected to be high simultaneously

$$G - Mean = \sqrt{Sensitivity \times Specificity}$$

Average Cost(AC) is expressed as

$$Average\ Cost = \frac{C_1 \times FN + C_{-1} \times FP}{TP + TN + FP + FN}$$

where C_1 and C_{-1} are the two costs used in the objective function of CS-SVM.

Youden's index Y equally weights the algorithm's performance on positive and negative instances:

$$Y = sensitivity + specificity - 1$$

Balanced accuracy(BA) is the average of sensitivity and specificity:

$$Balanced\ accuracy = \frac{sensitivity + specificity}{2}$$

4 Multi-objective optimization problems and Genetic algorithms

Multi-objective optimization problems consist of more than one criterion, often conflicting, for which any solution existing on the Pareto front of criterion trade-offs is considered optimal.

In this section, we introduce multi-objective optimization problems and the cornerstone concept of Pareto optimality.

A multi-objective problem consists of minimizing and/or maximizing two or more objective functions subject to

inequality and/or equality constraints. The objective functions are conflicting among them and a solution is a trade-off in the objective function space.

Definition 1 A solution is defined Pareto optimal if there does not exist any other solution in the objective space which improves the value of any of its objective functions without deteriorating at least one other objective function value.

In other words, a non-dominated solution provides a suitable compromise between all objectives without degrading any of them. The multi-objective optimization process is looking for a set of alternative solutions that represent the Pareto optimal solution. A set of non-dominated individuals form a Pareto-optimal front.

From the mathematical point of view, the definition of the dominance between two solutions x_1 and x_2 is that x_1 is no worse than x_2 in all objectives $f_i, i \in \{1, \ldots, m\}$ of the problem. This concept can be expressed as x_1 dominates x_2 if $f_i(x_1) \leq f_i(x_2)\ \forall i \in \{1, \ldots, m\}\ and\ \exists\ j \in \{1, \ldots, m\} : f_j(x_1) \leq f_j(x_2)$.

The genetic algorithms were developed by Holland and his collaborators (Holland 1975) as a model based on Charles Darwin's theory of natural selection. They are heuristic search techniques, successfully applied to different domains (e.g., Guido and Conforti 2017; Bao-De et al. 2021). Furthermore, they demonstrated a large amount of inherent parallelism that makes them attractive mainly for solving problems defined in large feature spaces, as that one here addressed. The evolutionary process usually starts from a population of randomly generated individuals, which are the chromosomes. It is an iterative process. One iteration is one generation. In each generation, the fitness of every individual in the population is evaluated. The fitness value of a chromosome is a measure of its goodness. The fitness is usually the value of the objective function in the optimization problem being solved. Usually, operators such as selection, crossover, mutation and recombination are applied during the evolutionary process over the generated populations to find better chromosomes, which optimize the fitness function till a termination condition is reached. The offsprings in a population act like independent agents so that they explore the search space in many directions.

As well known, genetic algorithms have some disadvantages mainly due to the choice of parameters such as the mutation rate and crossover rate that should be carried out carefully. The crossover operator is one of the most important operators because it determines the global convergence of the genetic algorithm.

4.1 NSGA-II

Srinivas and Deb (1994) proposed an algorithm based on non-dominated sorting for solving multiobjective problems. This algorithm was called non-dominated sorting genetic-algorithm (NSGA). Deb et al. (2002) improved it by proposing NSGA-II. The key features of NSGA-II are elitism, diversity-preserving mechanisms, and emphasis on non-dominated solutions. In NSGA-II, the N offsprings are created from the N parents using standard genetic algorithms. The new population at the next generation is given by selecting the non-dominated solutions for the Pareto front with the highest diversity while discarding the rest of the solutions.

Tournament selection This is a procedure that imitates survival of the fittest in nature. Indeed, each individual competes in two tournaments with randomly selected individuals. The crowded tournament selection is based on ranking and distance: if a solution has a better rank than another one, it will be selected; if the ranks are the same but the crowding distance is not, the solution with better crowding distance is selected.

Crowding distance The crowding distance metric of an individual proposed by Deb and Goel (2001) aims to select potential individuals to construct a new population. It is essentially based on the cardinality of a solution sets and their distance to solution boundaries. More specifically, it is defined as the perimeter of the rectangle with its nearest neighbors at diagonally opposite corners. Two individuals with a same rank are better if they have a larger crowding distance.

Crossover and mutation Crossover and mutation are employed to obtain the offspring population.

Algorithm 1 shows the framework of NSGA-II. The main steps of NSGA-II can be summarized as follows:

Step 1 Create a new population by combining parents and offsprings and apply non-dominated sorting
Step 2 Identify different fronts
Step 3 Generate the new population by exploiting the fronts given at the previous step until size N
Step 4 Use the crowd distance to carry out a crowding sort applied to the fronts
Step 5 Generate new offspring from the current population via the genetic operators crossover, mutation, and selection

5 Proposed approach

In this section, is firstly introduced a basic approach for hyper-parameter optimization. Then, a novel algorithm for hyper-parameters tuning based on GA and DT is proposed. The core of the algorithm is a fitness function evaluation procedure along with a similarity procedure.

Algorithm 1 NSGA-II

Require:

Random population P_0; a child population Q_0 is generated from the population of parents P_0 using genetic operators such as crossover and mutation

1: **while** any stopping criterion is not reached **do**
2: $R_t = P_t \cup Q_t$
3: fast-non-dominated-sort (R_t)
4: $P_{t+1} = \emptyset$; $i = 1$;
5: **while** $|P_{t+1}|+|F_i| < N$ **do**
6: Apply crowding-distance-assignment F_i
7: $P_{t+1} \leftarrow P_{t+1} \cup F_i$
8: $i \leftarrow i + 1$
9: **end while**
10: Sort ($F_i, < N$)
11: $P_{t+1} \leftarrow P_{t+1} \cup F_i[1 : (N - |P_t + 1|)]$
12: $Q_{t+1} \leftarrow$ create NewPop P_{t+1}
13: $t \leftarrow t + 1$
14: **end while**

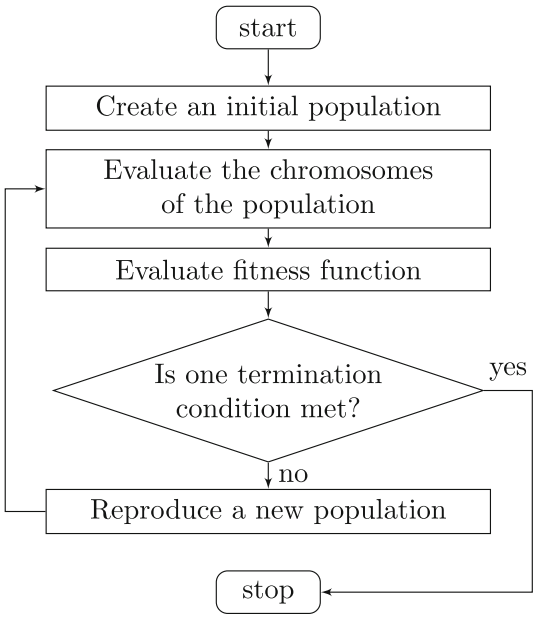

Fig. 1 Main steps of NSGA-II

5.1 Basic approach

The basic approach consists on using NSGA-II algorithm for solving a multi-objective hyper-parameter tuning problem. A set of hyper-parameter codified as a chromosome is evaluated by a k-fold CV approach. A fitness function evaluation is thus performed at each generation, i.e., each chromosome has its fitness functions evaluated. However, this approach is quite time consuming. Indeed, let N, be the number of chromosomes of a population, and G the number of generations. At each generation, the number of carried out k-fold CV is N, one per each chromosome. The overall number of performed k-fold CV is thus $N \times G$. For example, if $N = 24$ and $G = 200$, the overall number of k-fold CV to be carried out is 4800 and the computational time may be extremely high.

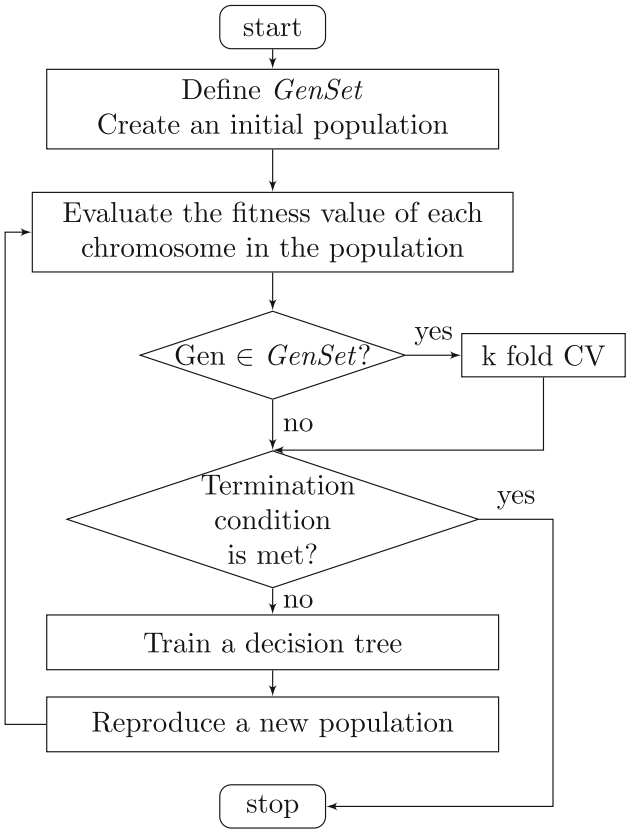

Fig. 2 Framework of the improved hyper-parameters algorithm

There are two main issues: the first one is related to the time needed to carry out k-fold CV; the second one, is related to the fact that often a chromosome is slightly different from another one already evaluated and with equal fitness. We try to overcome these two issues by introducing a procedure in the NSGA-II algorithm that exploits a suitable and trained DT. The proposed algorithm, described in the following, reduces considerably the overall number of performed k-fold CV by combining NSGA-II with a DT. The goal is to evaluate only a small set of chromosomes at each generation by a k-fold CV. This procedure does not affect convergence of the algorithm and strongly reduces the overall computational time.

5.2 Improved hyper-parameters algorithm

The above basic approach has been modified in order to evaluate the fitness function only of some individuals of a population by a k-fold CV. Figure 2 provides an intuitive understanding of the proposed algorithm framework.

Each chromosome consists of a number of genes that represent the hyper-parameters of CS-SVM. The algorithm

starts from an initial population Pop_0. It consists of the following five main steps.

Algorithm 2 Proposed hyper-parameters algorithm

1: Step 1 Initialization
2: Step 1.1 *Define* $GenSet$ as a set of numbers of generations
3: Step 1.2 *Create* an initial population Pop_0
4: Step 2 (Fitness function evaluation) *Evaluate* the fitness value of each chromosome in the current population.
5: **if** the current generation $Gen \in GenSet$ **then** go to Step 2.1
6: **else** go to Step 2.2
7: **end if**
8: Step 2.1 *Perform* a k-CV
9: Step 2.2 (Similarity procedure) *Compare* each chromosome with the ones of the previous population
10: **if** Similarity= True **then** assign a fitness value to it by the trained DT
11: **else** go to Step 2.1
12: **end if**
13: Step 3 *Termination criteria*. If at least one of the stopping conditions is meet, the algorithm stops
14: Step 4 *Train* Decision Tree. The current population is used to train a Decision Tree.
15: Step 5 *Reproduce* a new population. The operators of selection, crossover and mutation are applied over the generated population to find better chromosomes.

The core of Algorithm 2 is the fitness evaluation procedure at Step 2, explained in the following.

Step2: *Fitness evaluation procedure* The aim of the fitness evaluation step is to provide a procedure that reduces the number of fitness evaluations and consequently the number of carried out k-fold CV. To this purpose, a DT is trained at each generation and used to predict the fitness value of some chromosomes, as explained below. Indeed, the fitness of a chromosome in a population is evaluated or assigned: A whole population is evaluated by k-fold CV only at those generations well-defined in the set $GenSet$. This means that the cost-sensitive learning classifier SVM-based is built using the hyper-parameters codified as chromosomes of the population; for every chromosomes, a k-fold CV is used to estimate the generalization ability of the related build model. The set $GenSet$ has at least two elements, i.e., the first and the last generation. A procedure based on a learned DT takes place at those generations not in the set $GenSet$.

The fitness evaluation procedure is depicted in Fig. 3. To reduce the overall computational time, the procedure verifies if each chromosome has already a fitness value (because it has been evaluated previously). If so, the procedure analyzes next chromosome; otherwise, the chromosome is compared, at Step 2.2, with the chromosomes of the previous population in order to discover similarity. If the chromosome is similar at least to one chromosome, the DT trained on the previous population predicts its fitness value; this value is thus assigned as predicted value. Otherwise, if no similarity

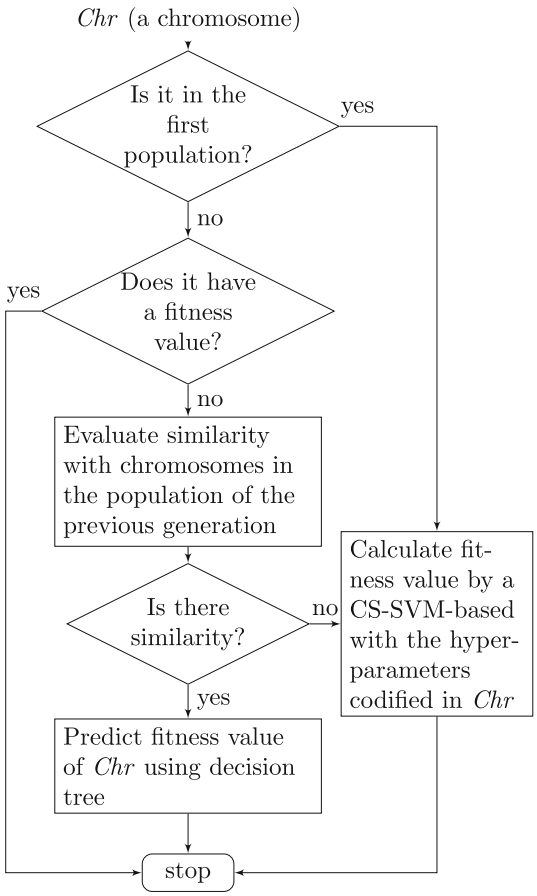

Fig. 3 Fitness evaluation procedure

is found, a cost-sensitive learning classifier SVM-based is built and the fitness value is evaluated by k-fold CV.

Similarity between two chromosomes can be estimated by various distance measurement methods. Here, we designed a procedure that evaluates similarity between two chromosomes as follows. Let chr_1 and chr_2, be two chromosomes represented as vectors. The procedure compares each corresponding couple of genes of chr_1 and chr_2, as detailed in Algorithm 3. More specifically, the difference between the $i-th$ gene of chr_1 and the corresponding gene of chr_2 is computed. If this difference is less than a given threshold t_i, the next couple of genes of the two chromosomes are compared; otherwise, the procedure stops and the two chromosomes are not similar.

Figure 4 depicts an example of DT trained to predict a given fitness function.

6 Experimental results and analysis

In this study, we test the proposed Algorithm 2 for on six benchmark imbalanced datasets binary classification task to compare the performance of different classification methods

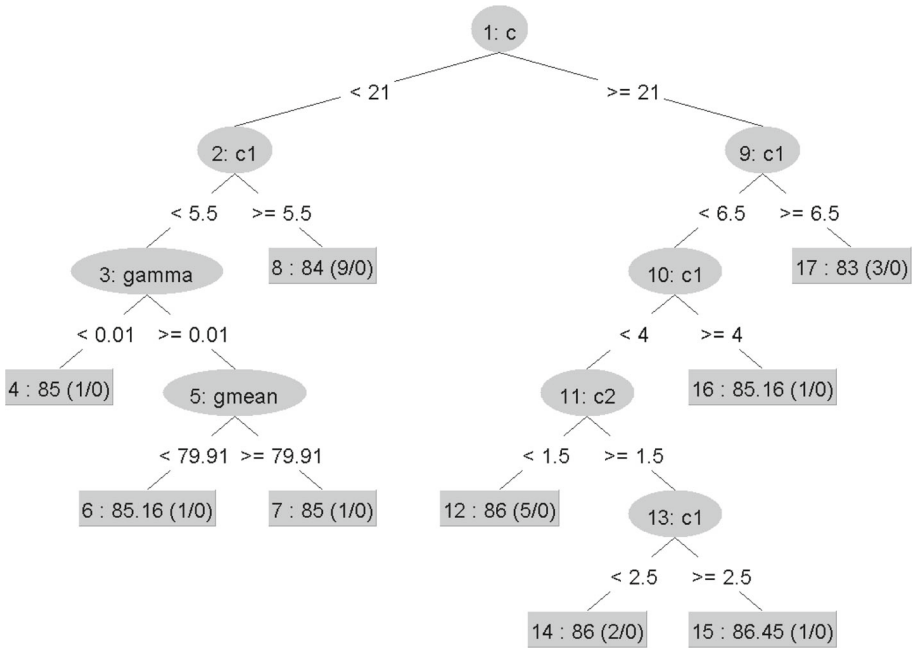

Fig. 4 An example of trained decision tree

Algorithm 3 Similarity procedure

Require:
Two chromosomes $chr_1, chr_2 \in R^k$. Threshold $t_i, i = 1, \ldots, k$.
1: $i = 1$; $similarity \leftarrow true$
2: **while** $i \leq k$ **do**
3: **if** $|chr_{i1} - chr_{i2}| < t_i$ **then**
4: $i \leftarrow i + 1$
5: **else**
6: $i \leftarrow k$
7: $similarity \leftarrow false$
8: **end if**
9: **end while**
10: **return** $similarity$

in the literature with our results. They are related to medical diagnosis represented as binary classification problems and have different sample sizes, attributes, and imbalance ratio (IR), defined as m/M (Amin et al. 2016), where m is the number of the minority instances and M is the number of majority instances.

We conducted experiments to answer the following research questions empirically:

1. Does multi-objective optimization find much sparser solutions without a major loss in predictive performance compared to single-objective optimization?
2. Are there alternative metrics to the accuracy?
3. May the computational time be reduced by a machine learning technique?

A brief description of the datasets is in Sect. 6.1. Details on the algorithms embedded in our approach and the hyper-parameter spaces of the several CS-SVM that are being investigated and tuned over are reported in Sect. 6.2. Experimental results are listed in Sect. 6.3.

6.1 Benchmark datasets

The datasets are from the University of California Irvine (UCI) Repository of Machine Learning Databases (https://archive.ics.uci.edu/ml/datasets.php). They have diversity in the number of attributes and imbalance ratio. Moreover, the datasets have both continuous and categorical attributes, and some of them have missing values.

Appendicitis dataset consists of 106 instances and 8 attributes. The attributes are results of laboratory test.

Haberman dataset describes the five-year or greater survival of breast cancer patients. The study was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital. The dataset consists of 306 instances and 4 attributes. The outcome is patient survival. There are no missing values.

Hepatitis dataset is used to classify patients with hepatitis in the two classes, live or die. It consists of 155 instances and 19 attributes, 14 nominal attributes and 6 multi-valued attributes. It requires the determination of whether patients with hepatitis will either live or die. The problem aims to predict the presence or absence of hepatitis by using the results of various medical tests carried out on a patient. The dataset has missing values.

Pima Indian Diabetes dataset is used to predict whether or not a patient has diabetes. All patients are female, are at least 21 years old, and are of Pima Indian heritage. It has 8 laboratory features.

Table 2 Datasets and their main characteristics in terms of number of attributes (No. A), number of the minority instances (m), number of the majority instances (M), index ratio $IR = m/M$

Dataset	No. A	m	M	IR
Appendicitis	8	21	85	0.25
Haberman	4	81	225	0.36
Hepatitis	19	70	85	0.82
Pima	9	268	500	0.53
WDBC	10	241	458	0.53
WPBC	32	47	151	0.33

Wisconsin Diagnostic Breast Cancer (WDBC) dataset consists of 30 features computed by digitized image of fine needle aspirate of a breast mass index. The problem aims to predict whether or not the patient has breast cancer.

Wisconsin Prognostic Breast Cancer (WPBC) dataset has 198 instances that represent follow-up data for one breast cancer case, only those cases exhibiting invasive breast cancer and no evidence of distant metastases at the time of diagnosis. It is used in this paper to classify patients as recurrences before 24 months (positive class) or non-recurrence beyond 24 months (negative class). We removed the feature named "Time" from the dataset because it is the recurrence time for instances in the positive class and the disease-free time for the instances of the negative class.

Table 2 summarizes, per each dataset, the number of attributes, the number of minority instances (diseased examples), the number of the majority instances (non-diseased examples), and the index ratio.

6.2 Learning algorithms and hyperparameters optimization

We considered several model classifiers CS-SVM with Gaussian kernel tuned by the optimization algorithm proposed. The experiments were performed by the ML algorithms of Waikato Environment for Knowledge Analysis (WEKA). WEKA is an open-source collection of ML algorithms and data processing tools. We used Sequential minimal optimization algorithm for SVM and Random Tree algorithm for DT. For that concerning NSGA-II algorithm, we used the framework named Java Class Library for Evolutionary Computation (JCLEC) Ramírez et al. (2015, 2019), which is a Java suite for solving multi-objective optimization problems using evolutionary algorithms.

Algorithm 2 has been coded in Java using the NSGA II algorithm of the JCLEC framework. We executed both the sequential and the parallel version of the NSGA-II. The parallel version is more efficient since it performs function evaluations of different individuals in parallel. The experiments were run on a PC Intel Xeon E5 1620 CPUs with 4 cores at 3.50 GHz and 32 GB RAM.

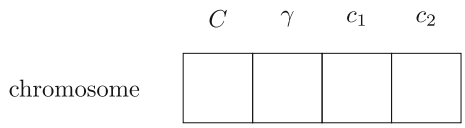

Fig. 5 Representation of a chromosome

6.2.1 Parameter setting

Algorithm 2 starts from an initial population Pop_0 of chromosomes randomly generated. Each chromosome has four genes representing the hyper-parameter of CS-SVM, as depicted in Fig. 5. All experiments have the same random initial population; the number of generations is the only one stopping criterion.

Table 3 lists the search population size, crossover probability p_c, gene mutation probability p_m, number of generations along with the design parameters (decision variables) and the range of their variations. We tested two population sizes and created the initial parent population randomly by selecting solutions from the ranges defined for the parameters C, C_1, C_2, γ, where C_1 and C_2 are the costs of the minority class and majority class, respectively.

The multi-objective problem that we formulated and solved has three fitness functions (7–9), given by accuracy, G-mean, and Average Cost, respectively:

$$f_1 = \max \frac{TP + TN}{TP + FP + TN + FN} \tag{7}$$

$$f_2 = \max \sqrt{Sensitivity \times Specificity} \tag{8}$$

$$f_3 = \min \frac{C_1 \times FN + C_2 \times FP}{TP + TN + FP + FN} \tag{9}$$

All the experiments are conducted by 10-fold cross-validation.

6.3 Computational results

To assess our approach, we performed both Algorithm 1 and Algorithm 2 on the six datasets and compared the results. The computational experiments were carried out using the JCLEC sequential algorithm and its parallelized version. The only difference we noticed was the reduced computational time of the parallelized version with respect to the sequential algorithm. We report in this section only the results found by the parallel version.

Table 4 reports the best fitness values per each dataset. From the second to the fourth column there is the value of accuracy, G-Mean, and average cost, respectively. We compare in this table our results with the best ones of the literature by selecting those papers that optimized hyper-parameter of

Table 3 NSGA-II parameters and hyper-parameters spaces of CS-SVM with RBF kernel

NSGA-II parameters				CS-SVM hyper-parameter space	
$Popsize$	p_c	p_m	Num Gen	Cost	γ
24	0.8	0.25	100	$C \in \{1-50\}$	$\{0.001-1\}$
48			200	$C_1 \in \{1-20\}$	
			1000	$C_2 \in \{1-10\}$	

Table 4 Best metric values by the optimized hyper-parameters compared to the best results of the literature. [1] (Yu and Wang 2017); [2] (Qi et al. 2013); [3](Tao et al. 2019). Hyphen means that the authors did not test that dataset

Dataset	Accuracy	G-Mean	AC	Acc [1]	AC [2]	G-Mean^a [3]	G-Mean^b [3]
Appendicitis	89.62	82.54	0.11	–	–		
Haberman	76.14	67.70	0.26	–	–	60.77±3.89	66.71±1.67
Hepatitis	**87.10**	84.06	0.14	83.22	0.208		
Pima	**78.13**	76.54	0.22	76.27	0.457	64.60±3.16	75.13±1.67
WDBC	97.42	97.73	0.03	–	–	92.41±2.44	96.91±1.79
WPBC	77.78	61.54	0.22	**81.28**	–	27.38±11.69	67.53±3.71

Bold indicates the best accuracy values

SVM with RBF kernel. Under these conditions, experimental evidence shows that our algorithm finds similar results or outperforms the other algorithms proposed in the literature. The best results in terms of accuracy on Hepatitis, Pima, and WPBC datasets are found in (Yu and Wang 2017) by optimizing the parameters of the SVM with RBF kernel by a novel ensemble differential evolution approach that they proposed. Several approaches were tested in (Tao et al. 2019) and the results were reported in terms of G-Mean. In Table 4, we denoted with G-Mean^a and G-Mean^b the values found by CS-SVM and their self-adaptive cost weights-based support vector machine cost-sensitive ensemble approach, respectively. It is helpful to notice that they reported these results on the datasets by modifying imbalanced data ratio of 10:1.

Tables 5, 6, 7, 8 list only some of the found non-dominated solutions of the Pareto front of our experimental results. These results refer to the experiments carried out with the related parallelized version of Algorithms 1 and 2. The first and second column of these tables report the population size and the number of carried out generations; the next three columns show the fitness function values associated with the optimal hyper-parameter configuration, which is reported in the next four columns. The eleventh and twelfth column reports the sensitivity and specificity values, whereas the next four columns report the ROC area, the F-Measure, the balanced accuracy, and the Youden's index, respectively. Finally, the last column shows the average computational time, in minutes.

As already observed in the literature, the accuracy is not a suitable measure for imbalanced data. Indeed, we noticed that in the Haberman dataset, for instance, there is a hyper-parameter configuration that allows to have a good accuracy equal to 75.53%, but the specificity is zero as well as the G-mean value. Similar cases are on the Hepatitis and WPBC datasets.

Tables 5 and 6 show the results found by Algorithm 1 on the six datasets along with the related optimized hyper-parameter configuration. For the Appendicitis dataset, for instance, the best accuracy in Table 5 is $f_1 = 89.62$; the best G-Mean is $f_2 = 82.54$, and the best average cost is $f_3 = 0.11$. As expected, the improvement of a fitness function implies a worsening in the other two. We observe that the single optimal fitness values are found with different hyper-parameter tuning. Moreover, the best results are found in all the experiments even if number of population size and generation number is increased.

Tables 7 and 8 show the results found by Algorithm 2 on the six datasets along with the related optimized hyper-parameter configuration. These results were found in very contracted computational time if compared to the previous ones. Observe that there has been a reduction over 70% in some experiments. These results show that the proposed Algorithm 2 is efficient.

The results evidenced that: (1) both algorithms converge and find the same best values for the three fitness functions; (2) the number of optimal non-dominated solutions of the Pareto front found by Algorithm 1 is greater than the number found by Algorithm 2. To better understand our finding, we illustrate in Figs. 6 and 7 the Pareto points of Tables 7 and 8 per each dataset with the six performance measures. The Pareto points are shown considering decreasing Sensitivity. As depicted in these two figures, generally balance accuracy decreases as Sensitivity decreases while Sensitivity increases. The best Pareto points related to medical datasets, as those tested in this paper, should be the points with high balance accuracy or high sensitivity values.

Table 5 Algorithm 1: Experimental results on Appendicitis, Haberman, and Hepatitis datasets

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
Appendicitis	24	100	89.62	77.28	0.18	33	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	0.76
			86.79	82.54	0.27	9	0.38	1	4	0.762	0.894	0.828	0.7	0.828	0.656	
			88.68	74.25	0.11	45	0.18	1	1	0.571	0.965	0.768	0.67	0.768	0.536	
		200	89.62	77.28	0.18	2	0.18	1	2	0.619	0.965	0.792	0.7	0.792	0.584	1.42
			86.79	82.54	0.27	9	0.38	1	4	0.762	0.894	0.828	0.7	0.828	0.656	
			88.68	74.25	0.11	10	0.38	1	1	0.571	0.965	0.768	0.67	0.768	0.536	
		1000	89.62	77.28	0.18	2	0.18	1	2	0.619	0.965	0.792	0.7	0.792	0.584	7.09
			86.79	82.54	0.27	9	0.38	1	4	0.762	0.894	0.828	0.7	0.828	0.656	
			88.68	74.25	0.11	15	0.38	1	1	0.571	0.965	0.768	0.67	0.768	0.536	
	48	100	89.62	77.28	0.18	39	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	1.44
			89.62	74.7	0.1	39	0.12	1	1	0.571	0.976	0.774	0.69	0.773	0.547	
			86.79	82.54	0.27	10	0.4	1	4	0.762	0.894	0.828	0.7	0.828	0.656	
			88.68	81.48	1.3	3	0.01	5	18	0.714	0.929	0.822	0.71	0.8215	0.643	
		200	89.62	77.28	0.18	26	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	2.83
			89.62	74.7	0.1	39	0.12	1	1	0.571	0.976	0.774	0.69	0.773	0.547	
		1000	89.62	77.28	0.18	26	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	14.13
			89.62	74.7	0.1	39	0.12	1	1	0.571	0.976	0.774	0.69	0.773	0.547	
			86.79	82.54	0.27	11	0.4	1	4	0.762	0.894	0.828	0.7	0.828	0.656	
Haberman	24	100	75.82	62.22	0.39	49	0.86	1	2	0.444	0.871	0.658	0.49	0.657	0.315	4.41
			70.92	65.7	2.19	49	0.53	4	13	0.568	0.76	0.664	0.51	0.664	0.328	
			73.86	34.43	0.26	48	0.74	1	1	0.123	0.96	0.542	0.2	0.541	0.083	
		200	75.82	62.22	0.39	46	0.86	1	2	0.444	0.871	0.658	0.49	0.657	0.315	9.02
			71.24	65.89	2.18	38	0.53	4	13	0.568	0.764	0.666	0.51	0.666	0.332	
			73.86	34.43	0.26	29	0.74	1	1	0.123	0.96	0.542	0.2	0.541	0.083	
		1000	75.82	62.22	0.39	46	0.86	1	2	0.444	0.871	0.658	0.49	0.657	0.315	37.96
			71.24	66.91	2.12	13	0.86	4	13	0.593	0.756	0.674	0.52	0.674	0.349	
			74.18	34.51	0.26	23	0.86	1	1	0.123	0.964	0.544	0.2	0.5435	0.087	

Table 5 continued

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
	48	100	76.14	62.38	0.39	49	0.94	1	2	0.444	0.876	0.66	0.5	0.66	0.32	6.78
			69.28	67.57	1.09	39	0.42	2	7	0.642	0.711	0.677	0.53	0.676	0.353	
			73.86	37.54	0.26	49	0.94	1	1	0.148	0.951	0.55	0.23	0.549	0.099	
		200	76.14	62.38	0.39	49	0.94	1	2	0.444	0.876	0.66	0.5	0.66	0.32	13.35
			69.28	67.57	1.09	39	0.42	2	7	0.642	0.711	0.677	0.53	0.676	0.353	
			73.86	37.54	0.26	49	0.94	1	1	0.148	0.951	0.55	0.23	0.549	0.099	
			74.18	32.81	0.26	47	0.52	1	1	0.111	0.969	0.54	0.19	0.54	0.08	
		1000	76.14	62.38	0.39	49	0.94	1	2	0.444	0.876	0.66	0.5	0.66	0.32	66.92
			72.55	67.7	2.07	2	0.94	4	13	0.593	0.773	0.683	0.53	0.683	0.366	
			73.86	37.54	0.26	49	0.94	1	1	0.148	0.951	0.55	0.23	0.549	0.099	
Hepatitis	24	100	86.45	80.54	0.33	40	0.01	3	2	0.719	0.902	0.811	0.69	0.810	0.621	0.94
			83.87	82.88	0.59	8	0.01	9	2	0.813	0.846	0.829	0.68	0.829	0.659	
			85.81	75.78	0.14	47	0.01	1	1	0.625	0.919	0.772	0.65	0.772	0.544	
		200	86.45	80.54	0.33	39	0.01	3	2	0.719	0.902	0.811	0.69	0.810	0.621	1.78
			83.87	82.88	0.59	8	0.01	9	2	0.813	0.846	0.829	0.68	0.829	0.659	
			85.81	75.78	0.14	47	0.01	1	1	0.625	0.919	0.772	0.65	0.772	0.544	
		1000	87.1	80.9	0.32	35	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	9.03
			83.87	84.06	0.29	8	0.01	5	1	0.844	0.837	0.841	0.68	0.840	0.681	
			85.81	75.78	0.14	47	0.01	1	1	0.625	0.919	0.772	0.65	0.772	0.544	
	48	100	87.1	80.9	0.32	32	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	1.82
			85.16	83.68	0.49	10	0.01	7	2	0.813	0.862	0.837	0.69	0.837	0.675	
			85.81	74.18	0.14	10	0.32	1	1	0.594	0.927	0.76	0.63	0.760	0.521	
		200	87.1	80.9	0.32	32	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	3.45
			85.16	83.68	0.49	10	0.01	7	2	0.813	0.862	0.837	0.69	0.837	0.675	
			85.81	75.78	0.14	47	0.01	1	1	0.625	0.919	0.772	0.65	0.772	0.544	
		1000	87.1	80.9	0.32	32	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	16.86
			83.87	84.06	0.29	8	0.01	5	1	0.844	0.837	0.841	0.68	0.8405	0.681	
			85.81	75.78	0.14	47	0.01	1	1	0.625	0.919	0.772	0.65	0.772	0.544	

Table 6 Algorithm 1: Experimental results on the datasets Pima, WDBC, and WPBC

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
Pima	24	100	77.99	72.69	1.46	5	0.24	7	6	0.604	0.874	0.739	0.66	0.739	0.478	10.19
			77.99	70.9	0.22	4	0.53	1	1	0.56	0.898	0.729	0.64	0.729	0.458	
			74.35	76.28	0.68	5	0.53	5	2	0.843	0.69	0.767	0.7	0.766	0.533	
			77.6	71.12	0.22	10	0.53	1	1	0.571	0.886	0.728	0.64	0.728	0.457	
		200	77.99	72.69	1.46	5	0.24	7	6	0.604	0.874	0.739	0.66	0.739	0.478	20.31
			77.99	70.9	0.22	4	0.53	1	1	0.56	0.898	0.729	0.64	0.729	0.458	
			74.35	76.28	0.68	5	0.53	5	2	0.843	0.69	0.767	0.7	0.766	0.533	
			77.6	71.12	0.22	10	0.53	1	1	0.571	0.886	0.728	0.64	0.728	0.457	
		1000	77.99	72.69	1.46	5	0.24	7	6	0.604	0.874	0.739	0.66	0.739	0.478	100.99
			77.99	70.9	0.22	4	0.53	1	1	0.56	0.898	0.729	0.64	0.729	0.458	
			74.87	76.54	0.99	4	0.53	7	3	0.832	0.704	0.768	0.7	0.768	0.536	
			77.86	71.28	0.22	15	0.53	1	1	0.571	0.89	0.73	0.64	0.730	0.461	
	48	100	78.13	70.82	0.22	4	0.47	1	1	0.556	0.902	0.729	0.64	0.729	0.458	21.26
			74.87	76.54	0.99	4	0.47	7	3	0.832	0.704	0.768	0.7	0.768	0.536	
			77.86	71.28	0.22	19	0.47	1	1	0.571	0.89	0.73	0.64	0.730	0.461	
		200	78.13	70.82	0.22	4	0.47	1	1	0.556	0.902	0.729	0.64	0.729	0.458	41.87
			74.87	76.54	0.99	4	0.47	7	3	0.832	0.704	0.768	0.7	0.768	0.536	
			77.86	71.28	0.22	19	0.47	1	1	0.571	0.89	0.73	0.64	0.730	0.461	
		1000	78.13	70.82	0.22	4	0.47	1	1	0.556	0.902	0.729	0.64	0.729	0.458	210.71
			78.13	72.34	2.11	4	0.47	10	9	0.593	0.882	0.738	0.65	0.737	0.475	
			74.87	76.54	0.99	4	0.47	7	3	0.832	0.704	0.768	0.7	0.768	0.536	
			77.99	71.51	0.22	18	0.47	1	1	0.575	0.89	0.732	0.65	0.732	0.465	
WDBC	24	100	97.42	97.73	0.06	10	0.42	2	5	0.988	0.967	0.977	0.96	0.9775	0.955	2.26
			97.14	97.7	0.03	6	0.42	1	5	0.996	0.959	0.977	0.96	0.9775	0.955	
		200	97.42	97.73	0.06	10	0.42	2	5	0.988	0.967	0.977	0.96	0.9775	0.955	7.49
			97.14	97.7	0.03	6	0.42	1	5	0.996	0.959	0.977	0.96	0.9775	0.955	
		1000	97.42	97.73	0.06	8	0.42	2	5	0.988	0.967	0.977	0.96	0.9775	0.955	30.76
			97.28	97.53	0.03	5	0.42	1	2	0.983	0.967	0.975	0.96	0.975	0.95	
	48	100	97.42	97.73	0.03	8	0.09	1	3	0.988	0.967	0.977	0.96	0.9775	0.955	9.44
		200	97.42	97.73	0.03	8	0.09	1	3	0.988	0.967	0.977	0.96	0.9775	0.955	11.8
		1000	97.42	97.73	0.03	8	0.09	1	3	0.988	0.967	0.977	0.96	0.9775	0.955	50.86
WPBC	24	100	77.78	29.08	0.24	38	0.24	1	4	0.085	0.993	0.539	0.15	0.539	0.078	1.65
			69.7	61.05	0.55	1	0.24	3	1	0.489	0.762	0.625	0.43	0.6255	0.251	
			76.77	14.59	0.23	38	0.24	1	5	0.021	1	0.511	0.04	0.5105	0.021	
		200	77.78	29.08	0.24	44	0.24	1	4	0.085	0.993	0.539	0.15	0.539	0.078	3.25
			77.78	25.26	0.22	32	0.42	1	8	0.064	1	0.532	0.12	0.532	0.064	
			69.7	61.05	0.55	1	0.24	3	1	0.489	0.762	0.625	0.43	0.6255	0.251	
		1000	77.78	29.08	0.24	18	0.42	1	4	0.085	0.993	150	0.15	0.539	0.078	16.17
			77.78	25.26	0.22	32	0.42	1	8	0.064	1	151	0.12	0.532	0.064	
			69.7	61.05	0.55	1	0.24	3	1	0.489	0.762	115	0.43	0.6255	0.251	

Table 6 continued

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
	48	100	77.78	25.26	0.22	47	0.33	1	9	0.064	1	0.532	0.12	0.532	0.064	3.15
			77.78	29.08	0.24	32	0.32	1	5	0.085	0.993	0.539	0.15	0.539	0.078	
			68.69	61.54	1.75	46	0.01	10	3	0.511	0.742	0.626	0.44	0.6265	0.253	
		200	77.78	29.08	0.24	33	0.32	1	5	0.085	0.993	0.539	0.15	0.539	0.078	6.34
			77.78	25.26	0.22	33	0.32	1	6	0.064	1	0.532	0.12	0.532	0.064	
			68.69	61.54	1.75	44	0.01	10	3	0.511	0.742	0.626	0.44	0.6265	0.253	
		1000	77.78	29.08	0.23	21	0.32	1	3	0.085	0.993	0.539	0.15	0.539	0.078	31.85
			77.78	25.26	0.22	33	0.32	1	6	0.064	1	0.532	0.12	0.532	0.064	
			68.69	61.54	1.75	48	0.01	10	3	0.511	0.742	0.626	0.44	0.6265	0.253	

Table 7 Algorithm 2: Experimental results on the datasets Appendicitis, Haberman, and Hepatitis

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
Appendicitis	24	100	89.62	77.28	0.18	23	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	0.37
			86.79	80.44	0.3	10	0.01	1	4	0.714	0.906	0.81	0.68	0.81	0.62	
			84.91	60.62	0.15	24	0.01	1	1	0.381	0.965	0.673	0.5	0.673	0.346	
		200	89.62	77.28	0.36	35	0.01	2	4	0.619	0.965	0.792	0.7	0.792	0.584	0.55
			86.79	82.54	0.32	28	0.13	1	5	0.762	0.894	0.828	0.7	0.828	0.656	
		1000	89.62	77.28	0.18	38	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	1.94
			87.74	80.96	0.29	8	0.01	1	4	0.714	0.918	0.816	0.7	0.816	0.632	
	48	100	89.62	77.28	0.36	29	0.01	2	4	0.619	0.965	0.792	0.7	0.792	0.584	0.61
			87.74	80.96	0.29	29	0.01	1	4	0.714	0.918	0.816	0.7	0.816	0.632	
			86.79	67.78	0.13	36	0.01	1	1	0.476	0.965	0.72	0.59	0.720	0.441	
		200	89.62	77.28	0.18	11	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	0.91
			86.79	82.54	1.19	9	0.47	4	18	0.762	0.894	0.828	0.7	0.828	0.656	
		1000	89.62	77.28	0.18	18	0.01	1	2	0.619	0.965	0.792	0.7	0.792	0.584	3.07
			87.74	80.96	0.29	8	0.01	1	4	0.714	0.918	0.816	0.7	0.816	0.632	
			84.91	57.05	0.15	22	0.01	1	1	0.333	0.976	0.655	0.47	0.654	0.309	
Haberman	24	100	76.14	62.38	0.39	42	1	1	2	0.444	0.876	0.66	0.5	0.66	0.32	1.82
			68.3	66.49	1.12	11	1	2	7	0.63	0.702	0.666	0.51	0.666	0.332	
			73.53	0	0.26	23	0.94	2	1	0	1	0.5	0	0.5	0	
		200	76.14	57.81	1.38	29	0.12	3	7	0.37	0.902	0.636	0.45	0.636	0.272	3.8
			67.32	66.29	1.13	3	0.94	2	7	0.642	0.684	0.663	0.51	0.663	0.326	
			75.82	62.22	0.39	42	0.94	1	2	0.444	0.871	0.658	0.49	0.657	0.315	
		1000	76.14	62.38	0.39	35	0.94	1	2	0.444	0.876	0.66	0.5	0.66	0.32	19.47
			70.92	67.21	1.6	11	0.94	3	10	0.605	0.747	0.676	0.52	0.676	0.352	
			73.86	37.54	0.26	49	0.94	1	1	0.148	0.951	0.55	0.23	0.549	0.099	
	48	100	76.14	62.38	0.39	49	0.94	1	2	0.444	0.876	0.66	0.5	0.66	0.32	4.29
			68.95	67.35	1.09	15	0.63	2	7	0.642	0.707	0.674	0.52	0.674	0.349	
			73.86	37.54	0.26	49	0.94	1	1	0.148	0.951	0.55	0.23	0.549	0.099	
		200	76.14	62.38	0.39	49	0.94	1	2	0.444	0.876	0.66	0.5	0.66	0.32	6.74
			69.28	67.57	1.09	24	0.63	2	7	0.642	0.711	0.677	0.53	0.676	0.353	
			73.86	34.43	0.26	49	0.56	1	1	0.123	0.96	0.542	0.2	0.541	0.083	

Table 7 continued

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
		1000	76.14	62.38	0.39	34	0.94	1	2	0.444	0.876	0.66	0.5	0.66	0.32	27.73
			69.61	67.33	2.7	9	1	5	17	0.63	0.72	0.675	0.52	0.675	0.35	
			73.53	0	0.26	10	0.94	7	1	0	1	0.5	0	0.5	0	
Hepatitis	24	100	87.1	79.47	0.45	28	0.01	4	3	0.688	0.919	0.803	0.69	0.803	0.607	0.4
			83.23	82.49	0.61	6	0.01	9	2	0.813	0.837	0.825	0.67	0.825	0.65	
			83.23	67.78	0.17	28	0.01	1	1	0.5	0.919	0.709	0.55	0.709	0.419	
		200	87.1	80.9	0.32	32	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	0.58
			83.23	82.49	0.61	6	0.01	9	2	0.813	0.837	0.825	0.67	0.825	0.65	
			84.52	71.89	0.15	32	0.01	1	1	0.563	0.919	0.741	0.6	0.741	0.482	
		1000	87.1	80.9	0.32	35	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	1.9
			83.87	84.06	0.29	8	0.01	5	1	0.844	0.837	0.841	0.68	0.840	0.681	
			79.35	0	0.21	8	0.01	1	3	0	1	0.5	0	0.5	0	
	48	100	86.45	80.54	0.33	13	0.02	3	2	0.719	0.902	0.811	0.69	0.810	0.621	0.74
			83.23	83.65	0.59	7	0.01	10	2	0.844	0.829	0.837	0.68	0.836	0.673	
			85.81	74.18	0.14	26	0.02	1	1	0.594	0.927	0.76	0.63	0.760	0.521	
		200	87.1	80.9	0.32	35	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	1.1
			81.94	81.68	0.71	15	0.01	11	2	0.813	0.821	0.817	0.65	0.817	0.634	
			84.52	71.89	0.15	34	0.01	1	1	0.563	0.919	0.741	0.6	0.741	0.482	
		1000	87.1	80.9	0.32	32	0.01	3	2	0.719	0.911	0.815	0.7	0.815	0.63	4.03
			83.87	84.06	0.29	8	0.01	5	1	0.844	0.837	0.841	0.68	0.840	0.681	
			84.52	71.89	0.15	34	0.01	1	1	0.563	0.919	0.741	0.6	0.741	0.482	

Table 8 Algorithm 2: Experimental results on the datasets Pima, WDBC, and WPBC

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
Pima	24	100	77.73	73.2	1.02	5	0.24	5	4	0.623	0.86	0.742	0.66	0.741	0.483	7.8
			74.35	76.28	0.68	5	0.53	5	2	0.843	0.69	0.767	0.7	0.766	0.533	
			77.08	70.34	0.23	23	0.32	1	1	0.56	0.884	0.722	0.63	0.722	0.444	
			77.47	70.26	0.23	23	0.24	1	1	0.552	0.894	0.723	0.63	0.723	0.446	
		200	77.99	70.9	0.44	4	0.53	2	2	0.56	0.898	0.729	0.64	0.729	0.458	12.41
			74.35	76.28	0.68	5	0.53	5	2	0.843	0.69	0.767	0.7	0.766	0.533	
			74.74	58.03	0.28	5	0.53	1	2	0.351	0.96	0.655	0.49	0.655	0.311	
		1000	77.47	70.89	0.45	48	0.24	2	2	0.567	0.886	0.727	0.64	0.726	0.453	52.83
			77.47	70.1	0.23	12	0.24	1	1	0.549	0.896	0.722	0.63	0.722	0.445	
			74.48	76.33	0.68	48	0.13	5	2	0.84	0.694	0.767	0.7	0.767	0.534	
	48	100	78.13	71.13	0.22	3	0.63	1	1	0.563	0.898	0.731	0.64	0.730	0.461	9.86
			74.35	76.28	0.68	3	0.58	5	2	0.843	0.69	0.767	0.7	0.766	0.533	
		200	78.13	71.13	0.22	3	0.63	1	1	0.563	0.898	0.731	0.64	0.730	0.461	15.34
			75.13	75.44	0.33	10	0.42	2	1	0.765	0.744	0.754	0.68	0.754	0.509	
			74.35	76.22	0.68	2	0.63	5	2	0.84	0.692	0.766	0.7	0.766	0.532	
		1000	78.13	71.13	0.22	3	0.63	1	1	0.563	0.898	0.731	0.64	0.730	0.461	62.73
			78.13	72.34	2.11	1	0.63	10	9	0.593	0.882	0.738	0.65	0.737	0.475	
			74.22	75.93	0.7	3	0.42	5	2	0.828	0.696	0.762	0.69	0.762	0.524	

Table 8 continued

Dataset	Gen. param.		Fitness functions			Optimized Hyper-par				Performance metrics						Time
	Pop size	Gen	f_1	f_2	f_3	C	γ	C_1	C_2	Sens	Spec	ROC area	F-M	BA	Y	
WDBC	24	100	97.42	97.73	0.09	19	0.24	3	7	0.988	0.967	0.977	0.96	0.977	0.955	0.87
			97.14	97.7	0.04	4	0.18	1	7	0.996	0.959	0.977	0.96	0.977	0.955	
		200	97.42	97.73	0.09	19	0.24	3	7	0.988	0.967	0.977	0.96	0.977	0.955	2.59
			97.28	97.43	0.04	4	0.01	1	3	0.979	0.969	0.974	0.96	0.974	0.948	
			97.14	97.7	0.04	4	0.18	1	7	0.996	0.959	0.977	0.96	0.977	0.955	
		1000	97.42	97.73	0.09	14	0.24	3	7	0.988	0.967	0.977	0.96	0.977	0.955	13.77
			97.14	97.7	0.03	14	0.24	1	4	0.996	0.959	0.977	0.96	0.977	0.955	
	48	100	97.42	97.73	0.09	19	0.24	3	7	0.988	0.967	0.977	0.96	0.977	0.955	2.52
			97.28	97.43	0.04	4	0.01	1	3	0.979	0.969	0.974	0.96	0.974	0.948	
			97.14	97.7	0.04	4	0.18	1	7	0.996	0.959	0.977	0.96	0.977	0.955	
		200	97.42	97.73	0.09	19	0.24	3	7	0.988	0.967	0.977	0.96	0.977	0.955	5.37
			97.28	97.43	0.04	4	0.01	1	3	0.979	0.969	0.974	0.96	0.974	0.948	
			97.14	97.7	0.04	4	0.18	1	7	0.996	0.959	0.977	0.96	0.977	0.955	
		1000	97.42	97.73	0.16	16	0.27	5	12	0.988	0.967	0.977	0.96	0.977	0.955	25.72
			97.14	97.13	0.03	4	0.27	1	1	0.971	0.972	0.971	0.96	0.971	0.943	
WPBC	24	100	76.26	0	0.24	49	0.01	1	1	0	1	0.5	0	0.5	0	0.96
			69.19	60.78	0.55	1	0.18	3	1	0.489	0.755	0.622	0.43	0.622	0.244	
		200	77.27	48.83	3.36	9	0.01	17	7	0.255	0.934	0.595	0.35	0.594	0.189	1.61
			65.15	62.29	2.95	5	0.01	17	5	0.574	0.675	0.625	0.44	0.624	0.249	
			76.26	0	0.24	42	0.01	1	7	0	1	0.5	0	0.5	0	
		1000	77.27	48.83	3.36	9	0.01	17	7	0.255	0.934	0.595	0.35	0.594	0.189	4.47
			65.15	62.29	2.95	5	0.01	17	5	0.574	0.675	0.625	0.44	0.624	0.249	
			76.26	0	0.24	5	0.01	1	9	0	1	0.5	0	0.5	0	
	48	100	76.26	0	0.24	41	0.09	1	2	0	1	0.5	0	0.5	0	1.77
			76.26	55.17	0.39	14	0.09	2	1	0.34	0.894	0.617	0.41	0.617	0.234	
		200	77.78	29.08	0.25	49	0.27	1	6	0.085	0.993	0.539	0.15	0.539	0.078	2.64
			77.78	25.26	0.22	48	0.27	1	7	0.064	1	0.532	0.12	0.532	0.064	
			67.17	60.71	3.04	47	0.01	17	5	0.511	0.722	0.616	0.42	0.616	0.233	
		1000	76.26	0	0.24	28	0.01	1	7	0	1	0.5	0	0.5	0	8.55
			65.15	59.59	3.25	37	0.01	18	5	0.511	0.695	0.603	0.41	0.603	0.206	

7 Conclusion

Support vector machines are one of the best ML models for solving several real-life classification problems. However, as in other ML techniques, their performance depends on hyper-parameters.

In this paper, we have investigated and proposed an approach that combines genetic algorithms and decision trees to optimize hyper-parameters of C-SVMs. The optimum values of the regularization parameter, costs of classes and the parameters of the RBF kernel function are searched for SVM.

We tested the algorithm on six benchmark datasets, which are imbalanced. We evaluated the performance of the models by several performance metrics. The framework is better or equivalent to other algorithms proposed in the literature for CS-SVM hyper-parameters optimization. Overall, taking into account three predictive metrics, i.e., accuracy, G-Mean, and average cost, the best hyper-parameter configuration is found in short computational time, mainly if compared with grid search approach. Hence, this approach can be considered as a good solution for addressing imbalanced dataset classification and hyper-parameter tuning, as they are challenging problems in classification research.

We suggest evaluating the performance of classifiers on medical data by suitable measures other than accuracy. Our future work is to extend and assess the proposed approach to investigate hyper-parameter tuning of different machine learning methods.

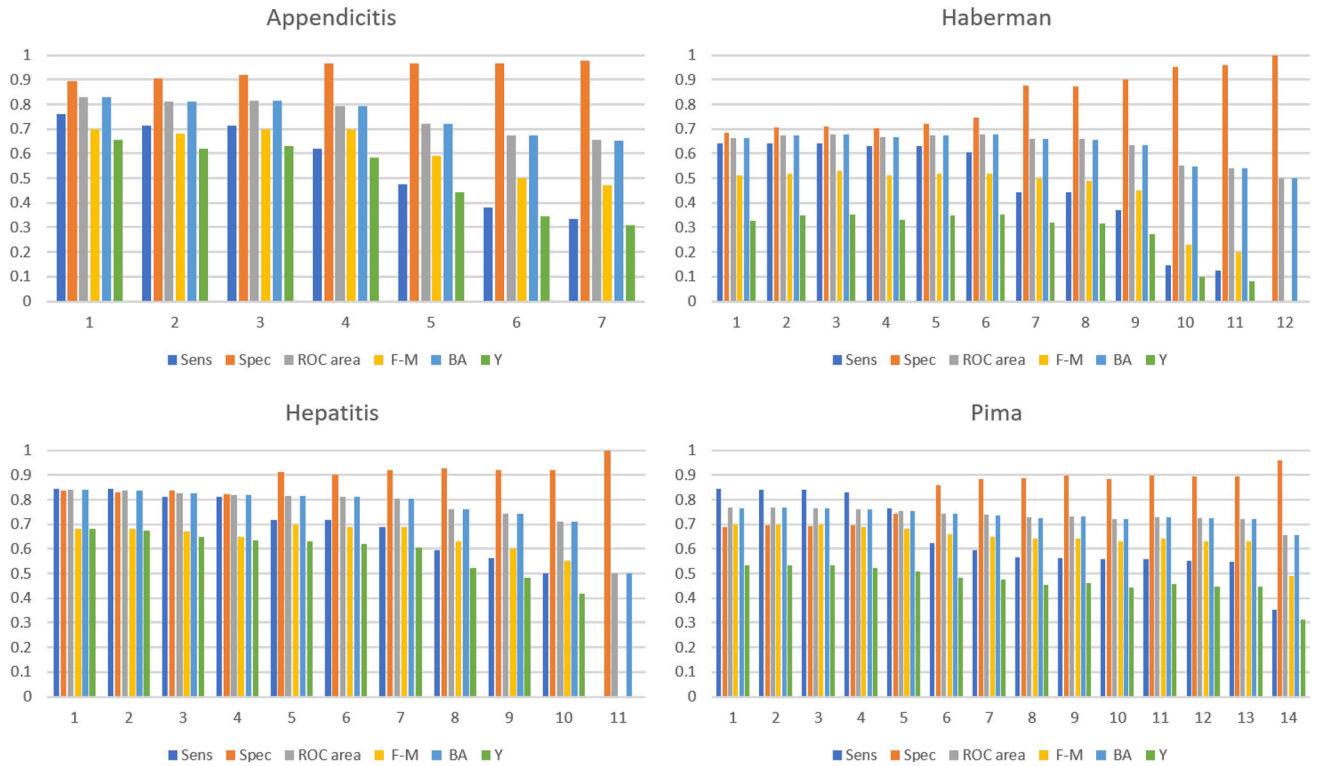

Fig. 6 Values of the six performance measures of the Pareto points found for Appendicitis, Haberman, Hepatitis, and Pima datasets

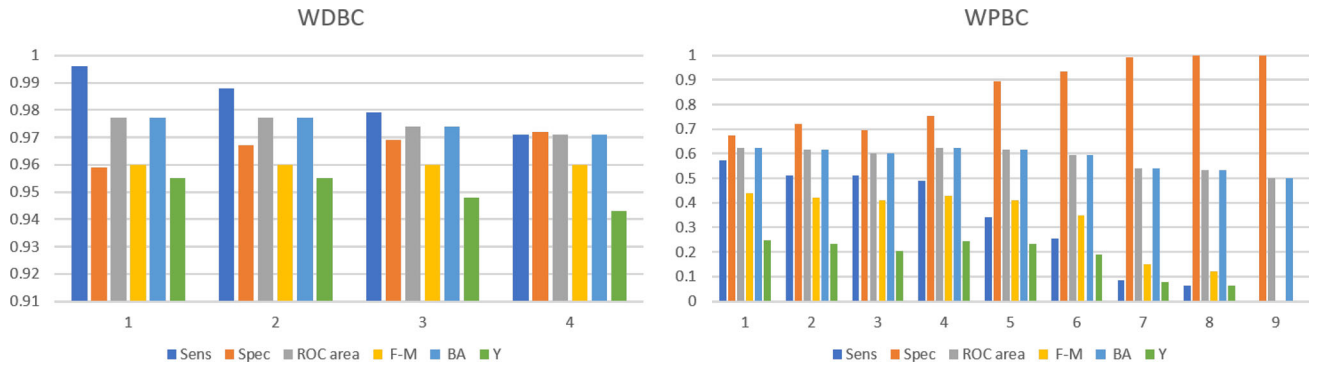

Fig. 7 Values of the six performance measures of the Pareto points found for WDBC and WPBC datasets

Acknowledgements The research has been partially supported by the research project SI.F.I.PA.CRO.DE. Sviluppo e industrializzazione farmaci innovativi per terapia molecolare personalizzata PA.CRO.DE. (PON *ARS*01_00568, CUP: B29C20000360005, CONCESSIONE RNA-COR: 4646672), Italian Ministry of University and Research, 2021.

Declaration

Conflict of interest The authors of the manuscript declare that they have no affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

Agrawal N, Kumar A, Bajaj V (2017) A new design method for stable IIR filters with nearly linear-phase response based on fractional derivative and swarm intelligence. IEEE Transactions on Emerging Topics in Computational Intelligence 1(6):464–477

Agrawal N, Kumar A, Bajaj V (2018) Design of digital IIR filter with low quantization error using hybrid optimization technique. Soft Comput 22(9):2953–2971

Amin A, Anwar S, Aea Adnan (2016) Comparing oversampling techniques to handle the class imbalance problem: a customer churn prediction case study. IEEE Access 4:7940–7957

Bao-De L, Xin-Yang Z, Mei Z et al (2021) Improved genetic algorithm-based research on optimization of least square support vector machines: an application of load forecasting. Soft Comput 10(1007):5674–9

Bergstra J, Bardenet R, Bengio Y, et al (2011) Algorithms for hyper-parameter optimization. In: and CAI (ed) Proceedings of the 24th international conference on neural information processing systems. USA, pp 2546–2554

Breiman L, Friedman JH, Olshen R, et al (1984) R. A. and Stone, C.J. Classification and regression trees. CRC press

Chawla N, Bowyer K, Lea Hall (2002) Smote: Synthetic minority oversampling technique. J Artif Intell Res 16:321–357

Cortes C, Vapnik V (1995) Support-vector network. Mach Learn 20:273–297

Cristianini N, Shawe-Taylor J (2000) An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press

Datta S, Das S (2015) Near-bayesian support vector machines for imbalanced data classification with equal or unequal misclassification costs. Neural Netw 70:39–52

Deb K, Goel T (2001) Controlled elitist non-dominated sorting genetic algorithms for better convergence. In: Lothar T, Kalyanmoy D, Coello C et al (eds) Zitzler Eckart. Evolutionary Multi-Criterion Optimization, Springer, Berlin Heidelberg, pp 67–81

Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197

Dwivedi AK, Ghosh S, Londhe ND (2018) Review and analysis of evolutionary optimization-based techniques for fir filter design. Circuits Syst Signal Process 37(10):4409–4430

Galar M, Fernandez A, Barrenechea E et al (2012) A review on ensembles for the class imbalance problem: Bagging, boosting, and hybrid-based approaches, systems, man, and cybernetics, part c: Applications and reviews. IEEE Trans 42(4):463–484

Goldberg DE, Holland J (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99

Guido R, Conforti D (2017) Hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. Comput Oper Res 87:270–282

Guido R, Groccia MC, Conforti D (2021) Hyper-Parameter Optimization in Support Vector Machine on unbalanced datasets using Genetic Algorithms. In: Optimization in Artificial Intelligence and Data Sciences, AIRO Springer Series (**in press**)

Hofmann T, Scholkopf B, Smola AJ (2008) Kernel methods in machine learning. Ann Statist pp 1171–1220

Holland JH (1975) Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence. Michigan Press

Iranmehr A, Masnadi-Shirazi H, Vasconcelos N (2019) Cost-sensitive support vector machines. Neurocomputing 343:50–64

Japkowicz N, Stephen S (2002) The class imbalance problem: a systematic study. Intell Data Anal 6:429–449

Jo T, Japkowicz N (2004) Class imbalances versus small disjuncts. ACM SIGKDD Explorations Newslett 6:40–49

Lessmann S, Stahlbock R, Crone R (2005) Optimizing hyperparameters of support vector machines by genetic algorithms. In: IC-AI pp 74–82

Mehrbakhsh N, Hossein A, Leila S et al (2019) A predictive method for hepatitis disease diagnosis using ensembles of neuro-fuzzy technique. J Infect Public Health 12(1):13–20

Noia A, Martino A, Montanari P et al (2020) Supervised machine learning techniques and genetic optimization for occupational diseases risk prediction. Soft Comput 24:4393–4406

Phienthrakul T, Kijsirikul B (2010) Evolutionary strategies for hyper-parameters of support vector machines based on multi-scale radial basis function kernels. Soft Comput 14:681–699

Qi Z, Tiana Y, Shia Y et al (2013) Cost-sensitive support vector machine for semi-supervised learning. Procedia Comput Sci 18:1684–1689

Ramírez A, Romero JR, Ventura S (2015) An extensible JCLEC-based solution for the implementation of multi-objective evolutionary algorithms. In: proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation, pp 1085–1092

Ramírez A, Romero JR, García-Martínez C et al (2019) JCLEC-MO: a java suite for solving many-objective optimization engineering problems. Eng Appl Artif Intell 81:14–28

Scholkopf B, Smola AJ (2001) Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA

Sokolova M, Japkowicz N, Szpakowicz S (2006) Beyond accuracy, F-score and ROC: A family of discriminant measures for performance evaluation. In: Sattar A, Kang B (eds) Advances in Artificial Intelligence. Lecture Notes in Computer Science, vol 4304. Springer, Berlin, Heidelberg

Srinivas N, Deb K (1994) Multiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2(3):221–248

Tao X, Li Q, Guo W et al (2019) Self-adaptive cost weights-based support vector machine cost-sensitive ensemble for imbalanced data classification. Inf Sci 487:31–56

Turney PD (1995) Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. J Artif Int Res 2:369–409

Vapnik V (1998) Statistical Learning Theory. Wiley, John Sons Inc

Veropoulos K, Campbell C, Cristianini N (1999) Controlling the sensitivity of support vector machines. In: proceedings of the international joint conference on AL, pp 55–60

Witten I, Frank E (2005) Data Mining Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, CA

Yu X, Wang X (2017) A novel hybrid classification framework using svm and differential evolution. Soft Comput 21:4029–4044