Combining Machine Learning and Metaheuristics Algorithms for Classification Method PROAFTN

Feras Al-Obeidat[1], Nabil Belacel[2(✉)], and Bruce Spencer[3]

[1] Zayed University, Abu-Dhabi, UAE
[2] Digital Technology Research Center, National Research Council, Ottawa, Canada
nabil.belacel@nrc-cnrc.gc.ca
[3] University of New Brunswick, Fredericton, Canada

Abstract. The supervised learning classification algorithms are one of the most well known successful techniques for ambient assisted living environments. However the usual supervised learning classification approaches face issues that limit their application especially in dealing with the knowledge interpretation and with very large unbalanced labeled data set. To address these issues fuzzy classification method PROAFTN was proposed. PROAFTN is part of learning algorithms and enables to determine the fuzzy resemblance measures by generalizing the concordance and discordance indexes used in outranking methods. The main goal of this chapter is to show how the combined meta-heuristics with inductive learning techniques can improve performances of the PROAFTN classifier. The improved PROAFTN classifier is described and compared to well known classifiers, in terms of their learning methodology and classification accuracy. Through this chapter we have shown the ability of the metaheuristics when embedded to PROAFTN method to solve efficiency the classification problems.

Keywords: Machine learning · Supervised learning · PROAFTN Metaheuristics

1 Introduction

In this chapter we introduce and compare various algorithms which have been used to enhance the performance of the classification method PROAFTN. It is a supervised learning that learns from a training set and builds set of prototypes to classify new objects [10,11]. The supervised learning classification methods have been applied extensively in Ambient Assisted Living (AAL) from sensors' generated data [36]. The enhanced algorithm can be used for instance to activity recognition and behavior analysis in AAL on sensors data [43]. It can be applied for the classification of daily living activities in a smart home using the generated sensors data [36]. Hence, the enhanced PROAFTN classifier can be integrated to active and assisted living systems as well as for smart homes

I. Ganchev et al. (Eds.): Enhanced Living Environments, LNCS 11369, pp. 53–79, 2019.
https://doi.org/10.1007/978-3-030-10752-9_3

health care monitoring frameworks as any classifiers used in the comparative study presented in this chapter [47]. This chapter is concerned with the supervised learning methods where the given samples or objects have known class labels called also training set, and the target is to build a model from these data to classify unlabeled instances called testing data. We focus on the classification problems in which classes are identified with discrete, or nominal, values indicating for each instance to which class it belongs, among the classes residing in the data set [21,60]. Supervised classification problems require a classification model that identifies the behaviors and characteristics of the available objects or samples called training set. This model is then used to assign a predefined class to each new object [31]. A variety of research disciplines such as statistics [60], Multiple Criteria Decision Aid (MCDA) [11,22] and artificial intelligence have addressed the classification problem [39]. The field of MCDA [10,63] includes a wide variety of tools and methodologies developed for the purpose of helping a decision model (DM) to select from finite sets of alternatives according to two or more criteria [62]. In MCDA, the classification problems can be distinguished from other classification problems within the machine learning framework from two perspectives [2]. The first includes the characteristics describing the objects, which are assumed to have the form of decision criteria, providing not only a description of the objects but also some additional preferential information associated with each attribute [22,51]. The second includes the nature of the classification pattern, which is defined in both ordinal, known as sorting [35], and nominal, known as multicriteria classification [10,11,63]. Classification based machine learning models usually fail to tackle these issues, focusing basically on the accuracy of the results obtained from the classification algorithms [62].

This chapter is devoted to the classification method based on the preference relational models known as outranking relational models as described by Roy [52] and Vincke [59]. The method presented in this paper employs a partial comparison between the objects to be classified and prototypes of the classes on each attribute. Then, it applies a global aggregation using the concordance and non-discordance principle [45]. Therefore it avoids resorting to conventional distance that aggregates the score of all attributes in the same value unit. Hence, it helps to overcome some difficulties encountered when data is expressed in different units and to find the correct preprocessing and normalization data methods. The PROAFTN method uses concordance and non-discordance principle that belongs to MCDA field developed by Roy [52,54]. Moreover, Zopounidis and Doumpos [63] dividing the classification problems based on MCDA into two categories: sorting problems for methods that utilize preferential ordering of classes and multicriteria classification for nominal sorting there is no preferential ordering of classes. In MCDA field the PROAFTN method is considered as nominal sorting or multicriteria classification [10,63]. The main characteristic of multicriteria classification is that the classification models do not automatically result only from the training set but depend also on the judgment of an expert. In this chapter we will show how techniques from machine learning and optimization can determine the accurate parameters for fuzzy the classification method

PROAFTN [11]. When applying PROAFTN method, we need to learn the value of some parameters, in case of our proposed method we have boundaries of intervals that define the prototype profiles of the classes, the attributes' weights, *etc.* To determine the attributes' intervals, PROAFTN applies the discretization technique as described by Ching *et al.* [20] from a set of pre-classified objects presenting a training set [13]. Even-though these approaches offer good quality solutions, they still need considerable computational time. The focus of this chapter concerns the application of different optimization techniques based on meta-heuristics for learning PROAFTN method. To apply PROAFTN method over very large data, there are many parameters to be set. If one were to use the exact optimization methods to infer these parameters, the computational effort that would be required is an exponential function of the problem size. Therefore, it is sometimes necessary to abandon the search for the optimal solution, using deterministic algorithms, and simply seek a good solution in a reasonable computational time, using meta-heuristics algorithms. In this paper, we will show how inductive learning method based on meta-heuristic techniques can lead to the efficient multicriteria classification data analysis.

The major characteristics of the multicriteria classification method compared with other well known classifiers can be summarized as follows:

- The PROAFTN method can apply two learning approaches: deductive or knowledge based and inductive learning. In the deductive approach, the expert has the role of establishing the required parameters for the studied problem for example the experts' knowledge or rules can be expressed as intervals, which can be implemented easily to build the prototype of the classes. In the inductive approach, the parameters and the classification models are obtained and learned automatically from the training dataset.
- PROAFTN uses the outranking and preference modeling as proposed by Roy [52] and it hence can be used to gain understanding about the problem domain.
- PROAFTN uses fuzzy sets for deciding whether an object belongs to a class or not. The fuzzy membership degree gives an idea about its weak and strong membership to the corresponding classes.

The overriding goal of this study is to present a generalized framework to learn the classification method PROAFTN. And then compare the performance and the efficiency of the learned method against well-known machine learning classifiers.

We shall conclude that the integration of machine learning techniques and meta-heuristic optimization to PROAFTN method will lead to significantly more robust and efficient data classification tool.

The rest of the chapter is organized as follows: Sect. 2 overviews the PROAFTN methodology and its notations. Section 3 explains the generalized learning framework for PROAFTN. In Sect. 4 the results of our experiments are reported. Finally, conclusions and future work are drawn in Sect. 5.

2 PROAFTN Method

This section describes the PROAFTN procedure, which belongs to the class of supervised learning to solve classification problems. Based on fuzzy relations between the objects being classified and the prototype of the classes, it seeks to define a membership degree between the objects and the classes of the problem [11]. The PROAFTN method is based on outranking relation as an alternative to the Euclidean distance through the calculation of an indifference index between the object to be assigned and the prototype of the classes obtained through the training phase. Hence, to assign an object to the class PROAFTN follow the rule known as concordance and no discordance principle as used by the outranking relations: if the object a is judged indifferent or similar to prototype of the class according to the majority of attributes "concordance principle" and there is no attribute uses its veto against the affirmation "a is an indifferent to this prototype" "no-discordance principal", the object a is considered indifferent to this prototype and it should be assigned to the class of this prototype [11,52].

PROAFTN has been applied to the resolution of many real-world practical problems such as acute leukemia diagnosis [14], asthma treatment [56], cervical tumor segmentation [50], Alzheimer diagnosis [18], e-Health [15] and in optical fiber design [53], asrtocytic and bladder tumors grading by means of computer-aided diagnosis image analysis system [12] and it was also applied to image processing and classification [1]. PROAFTN also has been applied for intrusion detection and analyzing Cyber-attacks [24,25]. Singh and Arora [55] present an interesting application of fuzzy classification PROAFTN to network intrusion detection. In this paper authors find that PROAFTN outperforms the well known classifier Support Vector Machine [55]. The following subsections describe the notations, the classification methodology, and the inductive approach used by PROAFTN.

2.1 PROAFTN Notations

The PROAFTN notations used in this paper are presented in Table 1.

2.2 Fuzzy Intervals

Let A represents a set of objects known as a training set. Consider a new object a to be classified. Let a be described by a set of m attributes $\{g_1, g_2, ..., g_m\}$. Let the k classes be $\{C^1, C^2, ..., C^k\}$. The different steps of the procedure are as follows:

For each class C^h, a set L_h of prototypes is determined. For each prototype b_i^h and each attribute g_j, an interval $[S_j^1(b_i^h),\ S_j^2(b_i^h)]$ is defined where $S_j^2(b_i^h) \geq S_j^1(b_i^h)$. Two thresholds $d_j^1(b_i^h)$ and $d_j^2(b_i^h)$ are introduced to define the fuzzy intervals: the pessimistic interval $[S_j^1(b_i^h), S_j^2(b_i^h)]$ and the optimistic interval $[S_j^1(b_i^h) - d_j^1(b_i^h), S_j^2(b_i^h) + d_j^2(b_i^h)]$. The pessimistic intervals are determined by applying discretization techniques from the training set as described in [26,28].

Table 1. Notations and parameters used by the PROAFTN method

A	Set of objects with known labels $\{a_1, a_2, ..., a_n\}$ the preassigned objects (training set)
$\{g_1, g_2, ..., g_m\}$	Set of m attributes:
Ω	set of k classes such as: $\Omega = \{C^1, C^2, ..., C^k\}$, $k \geq 2$
B^h	Prototype set of h^{th} category, where $B^h = \{b_i^h \mid h = 1, ..., k, i = 1, ..., L_h\}$
B	Set of all prototypes, such as $B = \bigcup_{h=1}^{k} B^h$
$[S_j^1(b_i^h),\ S_j^2(b_i^h)]$	The interval of the prototype b_i^h for each attribute g_j in class C^h with $j = 1, 2, ..., m$
$d_j^1(b_i^h)$ and $d_j^2(b_i^h)$	The preference thresholds belong to b_i^h for each attribute g_j in class C^h
w_{jh}	The weight of attribute g_j for the class C^h

The classical data mining techniques, such as decision tree, numerical domains "continuous numeric values" into intervals and the discretized intervals are treated as ordinal "discretized" values during induction. Ramírez-Gallego *et al.* [29] present more details on different approaches used for data discretization in machine learning. In our case the discretized intervals are treated as intervals and they are not treated as discrete value. As a result, PROAFTN avoids losing information in the induction process and also can use both inductive and deductive learning without transforming the continue values to discrete data. In deductive learning, the rules in our case can also be given by interacting with the expert in the form of ranges or intervals, and then can be optimized during the learning process. Figure 2 depicts the representation of PROAFTN's intervals. To apply PROAFTN, the pessimistic interval $[S_{jh}^1, S_{jh}^2]$ and the optimistic interval $[q_{jh}^1, q_{jh}^2]$ [13] of each attribute in each class need to be determined. Figure 2 depicts the representation of PROAFTN's intervals. When evaluating a certain quantity or a measure with a regular or crisp interval, there are two extreme cases, which we should try to avoid. It is possible to make a pessimistic evaluation, but then the interval will appear wider. It is also possible to make an optimistic evaluation, but then there will be a risk of the output measure to get out of limits of the resulting narrow interval, so that the reliability of obtained results will be doubtful. To overcome this problem we have introduced fuzzy approach to features' or criteria evaluation as presented in Fig. 1 [16]. They permit to have simultaneously both pessimistic and optimistic representations of the studied measure [23]. This is why we introduce the thresholds $d1$ and $d2$ for each attribute to define in the same time the both pessimistic interval $[S_j^1(b_i^h), S_j^2(b_i^h)]$ and the optimistic interval $[S_j^1(b_i^h) - d_j^1(b_i^h), S_j^2(b_i^h) + d_j^2(b_i^h)]$ [13]. The carrier of a fuzzy interval (from S1 minus d1 to S2 plus d2) will be chosen so that it guarantees not to override the considered quantity over necessary limits, and the kernel (S1 to S2) will contain the most true-like values [61]. To apply PROAFTN, the

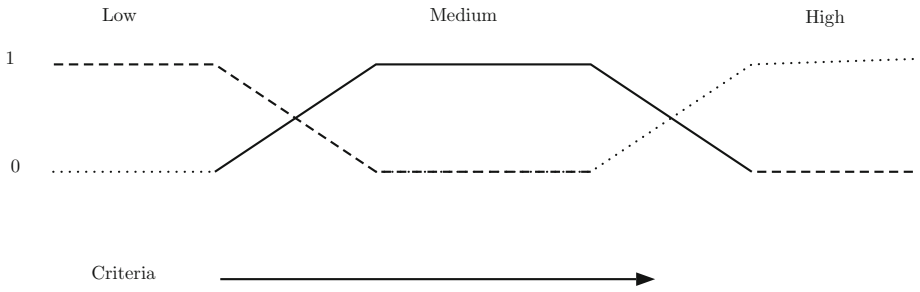

Fig. 1. Fuzzy approach for features evaluation

pessimistic interval $[S^1_{jh}, S^2_{jh}]$ and the optimistic interval $[q^1_{jh}, q^2_{jh}]$ [13] for each attribute in each class need to be determined, where:

$$q^1_{jh} = S^1_{jh} - d^1_{jh} \qquad q^2_{jh} = S^2_{jh} + d^2_{jh} \tag{1}$$

applied to:

$$q^1_{jh} \leq S^1_{jh} \qquad q^2_{jh} \geq S^2_{jh} \tag{2}$$

Hence, $S^1_{jh} = S^1_j(b^h_i)$, $S^2_{jh} = S^2_j(b^h_i)$, $q^1_{jh} = q^1_j(b^h_i)$, $q^2_{jh} = q^2_j(b^h_i)$, $d^1_{jh} = d^1_j(b^h_i)$, and $d^2_{jh} = d^2_j(b^h_i)$. The following subsections explain the stages required to classify the testing object a to the class C^h using PROAFTN.

2.3 Computing the Fuzzy Indifference Relation

The initial stage of classification procedure is performed by calculating the fuzzy indifference relation $I(a, b^h_i)$ or also called the fuzzy resemblance measure. The fuzzy indifference relation is based on the concordance and non-discordance principle which represents the relationship (membership degree) between the object to be assigned and the prototype [10,11]; it is formulated as:

$$I(a, b^h_i) = \left(\sum_{j=1}^{m} w_{jh} C^i_{jh}(a, b^h_i) \right) \prod_{j=1}^{m} \left(1 - D^i_{jh}(a, b^h_i)^{w_{jh}}\right) \tag{3}$$

where w_{jh} is the weight that measures the importance of a relevant attribute g_j of a specific class C^h:

$$w_{jh} \in [0, 1], \quad \text{and} \quad \sum_{j=1}^{m} w_{jh} = 1$$

$C^i_{jh}(a, b^h_i)$ is the degree that measures the closeness of the object a to the prototype b^h_i according to the attribute g_j.

$$C^i_{jh}(a, b^h_i) = \min\{C^1_{jh}(a, b^h_{i1}), C^{i2}_{jh}(a, b^h_i)\}, \tag{4}$$

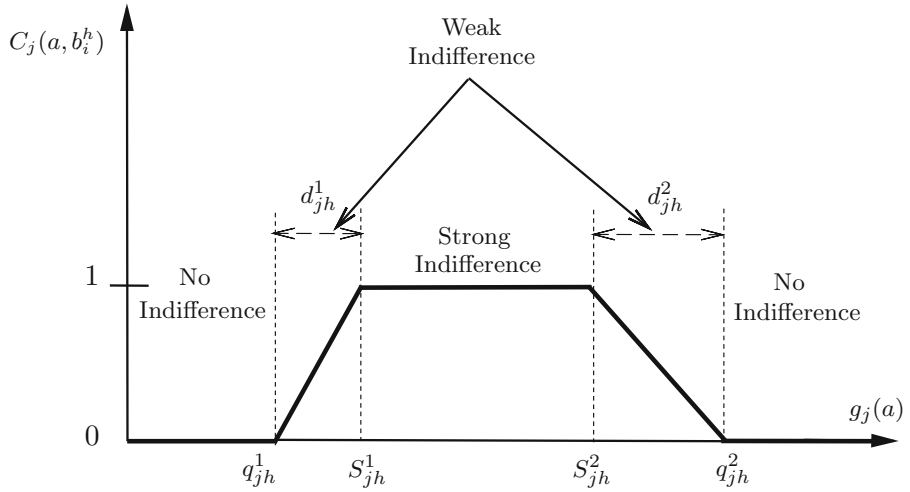

Fig. 2. Graphical representation of the partial indifference concordance index between the object a and the prototype b_i^h represented by intervals.

where

$$C_{jh}^{i1}(a, b_i^h) = \frac{d_j^1(b_i^h) - \min\{S_j^1(b_i^h) - g_j(a), d_j^1(b_i^h)\}}{d_j^1(b_i^h) - \min\{S_j^1(b_i^h) - g_j(a), 0\}}$$

and

$$C_{jh}^{i2}(a, b_i^h) = \frac{d_j^2(b_i^h) - \min\{g_j(a) - S_j^2(b_i^h), d_j^2(b_i^h)\}}{d_j^2(b_i^h) - \min\{g_j(a) - S_j^2(b_i^h), 0\}}$$

$D_{jh}^i(a, b_i^h)$, is the discordance index that measures how far the object a is from the prototype b_i^h according to the attribute g_j. Two veto thresholds $v_j^1(b_i^h)$ and $v_j^2(b_i^h)$ [11], are used to define this value, where the object a is considered perfectly different from the prototype b_i^h based on the value of attribute g_j. In general, the value of veto thresholds are determined by an expert familiar with problem. In this study the effect of the veto thresholds is not considered and only the concordance principle is used, so Eq. (3) is summarized by:

$$I(a, b_i^h) = \sum_{j=1}^{m} w_{jh} C_{jh}^i(a, b_i^h) \tag{5}$$

For more illustrations, the three comparative cases between the object a and prototype b_i^h according to the attribute g_j are obtained (Fig. 2):

- case 1 (strong indifference):
  $C_{jh}^i(a, b_i^h) = 1 \Leftrightarrow g_j(a) \in [S_{jh}^1, S_{jh}^2]$; (*i.e.*, $S_{jh}^1 \leq g_j(a) \leq S_{jh}^2$)
- case 2 (no indifference):
  $C_{jh}^i(a, b_i^h) = 0 \Leftrightarrow g_j(a) \leq q_{jh}^1$, or $g_j(a) \geq q_{jh}^2$

– case 3 (weak indifference):
The value of $C^i_{jh}(a, b^h_i) \in (0, 1)$ is calculated based on Eq. (4). (*i.e.*, $g_j(a) \in [q^1_{jh}, S^1_{jh}]$ or $g_j(a) \in [S^2_{jh}, q^2_{jh}]$)

The partial fuzzy indifference relation is represented by the trapezoidal membership function. This type of functions are well studied in the references [42] and [9]. Table 2 presents the performance matrix which is used to evaluate the prototype of classes on a set of attributes. The rows of the matrix represent the prototypes of the classes and the columns represent the attributes. The intersection between the row i and the column j corresponds to the partial indifference relation $C^i_{jh}(a, b^h_i)$ between the prototype b^h_i and the object a to be assigned according to the attribute g_j.

Table 2. Performance matrix of prototypes of the class C^h according to their partial fuzzy indifference relation with an object a to be classified.

	g_1	g_2	...	g_j	...	g_m
b^1_1	$C^1_{11}(a, b^1_1)$	$C^1_{21}(a, b^1_1)$	...	$C^1_{j1}(a, b^1_1)$	...	$C^1_{m1}(a, b^1_1)$
b^1_2	$C^2_{11}(a, b^1_2)$	$C^2_{21}(a, b^1_2)$	...	$C^2_{j1}(a, b^1_2)$	...	$C^2_{m1}(a, b^1_2)$
⋮	⋮	⋮	...	⋮	...	⋮
b^h_i	$C^i_{1h}(a, b^h_i)$	$C^i_{2h}(a, b^h_i)$	...	$C^i_{jh}(a, b^h_i)$	...	$C^i_{mh}(a, b^h_i)$
⋮	⋮	⋮	...	⋮	...	⋮
$b^k_{L_k}$	$C^{L_k}_{1k}(a, b^k_{L_k})$	$C^{L_k}_{2k}(a, b^k_{L_k})$	...	$C^{L_k}_{jk}(a, b^k_{L_k})$	...	$C^{L_k}_{mk}(a, b^k_{L_k})$

2.4 Evaluation of the Membership Degree

The membership degree $\delta(a, C^h)$ between the object a and the class C^h is calculated based on the indifference degree between a and its closest neighbor in the set of prototype B^h of the class C^h. To calculate the degree of membership of the object a to the class C^h, PROAFTN apply the formulae given by the Eq. 6.

$$\delta(a, C^h) = \max\{I(a, b^h_1), I(a, b^h_2), ..., I(a, b^h_{L_h})\} \tag{6}$$

2.5 Assignment of an Object to the Class

Once the membership degree of the testing "unlabeled" object a is calculated, the PROAFTN classifier will assign this object to the right class C^h by following the decision rule given by Eq. 7.

$$a \in C^h \Leftrightarrow \delta(a, C^h) = \max\{\delta(a, C^i)/i \in \{1, ..., k\}\} \tag{7}$$

3 Introduced Meta-heuristic Algorithms for Learning PROAFTN

The classification procedure used by PROAFTN to assign objects to the preferred classes is summarized in Algorithm 1.

Algorithm 1. PROAFTN classification procedure.

Input:A: set of objects; K: the number of classes; w_j^h the weight of the attribute j of the class h. A is divided into training and testing sets.
Output:$\delta(a, C^h)$: the membership degree of object a to class C^h
Step 1: Building the classification model for PROAFTN:
Assign a relative importance weights $w_j^h, j = 1, .., m; h = 1, ..., k$ to the attributes;
From the training set : Apply the discretization and inductive algorithm to build the prototype of the classes as in [7,16]. Each prototype b_h^i is defined by m attributes $g_j, j = 1, .., m$ with its score in each attribute is defined by two intervals: - pessimistic:$[S_j^1(b_i^h), S_j^2(b_i^h)]$; and - optimistic $[d_j^1(b_i^h), d_j^2(b_i^h)]$ as presented in Fig. 2.
Step 2: Compute the indifference relation between the object a and the prototype b_i^h of the class h:

$$I(a, b_i^h) = \sum_{j=1}^{m} w_j^h C_j(a, b_i^h) \tag{8}$$

$$C_j(a, b_i^h) = \min\{C_j^1(a, b_i^h), C_j^2(a, b_i^h)\}, \tag{9}$$

where

$$C_j^1(a, b_h^i) = \frac{d_j^1(b_i^h) - \min\{S_j^1(b_h^i) - g_j(a), d_j^1(b_i^h)\}}{d_j^1(b_i^h) - \min\{S_j^1(b_h^i) - g_j(a), 0\}},$$

$$C_j^2(a, b_h^i) = \frac{d_j^2(b_i^h) - \min\{g_j(a) - S_j^2(b_h^i), d_j^2(b_i^h)\}}{d_j^2(b_i^h) - \min\{g_j(a) - S_j^2(b_h^i), 0\}}$$

Step 3: Evaluation of the membership degree:

$$\delta(a, C^h) = \max\{I(a, b_1^h), I(a, b_2^h), ..., I(a, b_{L_h}^h)\} \tag{10}$$

Step 3: Assign the object a to the class:

$$a \in C^h \Leftrightarrow \delta(a, C^h) = \max\{\delta(a, C^i)/i \in \{1, ..., k\}\} \tag{11}$$

The rest of the chapter is to present the different methodologies based on machine learning and metaheuristic techniques for learning the classification method PROAFTN from data. The goal of the development of such methodologies is to obtain, from the training data set, the PROAFTN parameters that achieve the highest classification accuracy by applying the Algorithm 1. For this purpose, different learning methodologies are summarized in the following subsections.

3.1 Learn and Improve PROAFTN Based on Machine Learning Techniques

In [7,13], new methods were proposed to learn and improve PROAFTN based on machine learning techniques. The proposed learning methods consist of two stages: the first stage involves using a novel discretization technique to obtain the required parameters for PROAFTN, and the second stage is the development of a new inductive approach to construct PROAFTN prototypes for classification. Three unsupervised discretization methods – Equal Width Binning (EWB), Equal Frequency Binning (EFB) and k-Means – were used to establish PROAFTN parameters as described in algorithm. Algorithm 2 explains the utilization of discretization techniques and Chebyshev's theorem to obtain the parameters $\{S^1, S^2, d^1, d^2\}$ for PROAFTN. Firstly, the discretization technique is used to initially obtain the intervals $\{S^1_{jh}, S^2_{jh}\}$ for each attribute in each class. Secondly, Chebyshev's theorem is utilized to tune the generated intervals by discretization technique to obtain $\{d^1_{jh}, d^2_{jh}\}$ [16].

Algorithm 2. Developed techniques to obtain $\{S^1, S^2, d^1, d^2\}$

$z \leftarrow$ Number of classes
$m \leftarrow$ Number of attributes
$k \leftarrow$ Number of intervals (*i.e.,* number of clusters or bins)
for $h \leftarrow 1$, z **do**
  for $j \leftarrow 1$, m **do**
    Apply the discretization algorithm (k-Means, or EFB)
    The generated k clusters/bins represents the intervals' boundaries (*i.e.,* $\{S^1_{jh}, S^2_{jh}\}$:)
    Apply Chebyshev's on each interval to get $\{d^1_{jh}, d^2_{jh}\}$:
    for $r \leftarrow 1$, k **do**
      calculate the mean (μ) and the standard deviations (σ)
      for $t \leftarrow 2$, 5 **do**
        Calculate the ratio of values, which are between $\mu \pm t\sigma$
        if ratio $\geqslant (1 - 1/t^2)100$ **then**
          select $(\mu - t\sigma, \mu + t\sigma)$ as first interval i.e. Where:
          $S^{1r}_{jh} = \mu - t\sigma$, $S^{2r}_{jh} = \mu + t\sigma$
          $q^{1r}_{jh} = \mu - (t+1)\sigma$, $q^{2r}_{jh} = \mu + (t+1)\sigma$
          $d^{1r}_{jh} = S^{1r}_{jh}$ - q^{1r}_{jh} and $d^{2r}_{jh} = q^{2r}_{jh}$ - S^{2r}_{jh}
        end if
      end for
    end for
  end for
end for

Thereafter, an induction approach was introduced to compose PROAFTN prototypes to be used for classification. To evaluate the performance of the proposed approaches, a general comparative study was carried out between DT algorithms (C4.5 and ID3) and PROAFTN based on the proposed learning techniques. That portion of the study concluded that PROAFTN and DT algorithms

(C4.5 and ID3) share a very important property: they are both interpretable. In terms of classification accuracy, PROAFTN was able to outperform DT [16].

A superior technique for learning PROAFTN was introduced using Genetic algorithms (GA). More particularly, the developed technique, called GAPRO, integrates k-Means and a genetic algorithm to establish PROAFTN prototypes automatically from data in near optimal form. The purpose of using GA was to automate and optimize the selection of number of clusters and the thresholds to refining the prototypes. Based on the results generated by 12 typical classification problems, it was noticed that the newly proposed approach enabled PROAFTN to outperform widely used classification methods. The general description of using k-Means with GA to learn the PROAFTN classifier is documented in [7,13]. A GA is an adaptive metaheuristic search algorithm based on the concepts of natural selection and biological evolution. GA principles are inspired by Charles Darwin's theory of "survival of the fittest"; that is, the strong tend to adapt and survive while the weak tend to vanish. GA was first introduced by John H. Holland in the 1970s and further developed in 1975 to allow computers to evolve solutions to difficult search and combinatorial systems, such as function optimization and machine learning. As reported in the literature, GA represents an intelligent exploitation of a random search used to solve optimization problems. In spite of its stochastic behavior, GA is generally quite effective for rapid global searches for large, non-linear and poorly understood spaces; it exploits historical information to direct the search into the region of better performance within the search space [32,49].

In this work, GA is utilized to approximately obtain the best values for the threshold β and the number of clusters κ. The threshold β represents the ratio of the total number of objects from training set within each interval of each attribute in each class. As discussed earlier, to apply the discretization k-Means, the best κ value is required to obtain the intervals: $[S_j^1(b_i^h),\ S_j^2(b_i^h)]$, $[d_j^1(b_i^h),\ d_j^2(b_i^h)]$ and thresholds β as illustrated in Algorithm 4. In addition, the best value of β is also required to build the classification model that contains the best prototypes as described in Algorithm 4. Furthermore, since each dataset may have different values for κ and β, finding the best values for β and κ to compose PROAFTN prototypes is considered a difficult optimization task. As a result, GA is utilized to obtain these values. Within this framework, the value for β varies between 0 and 1 (*i.e.,* $\beta \in [0,1]$), and the value for κ changes from 2 to 9 ($\kappa \in 2,...,9$). The formulation of the optimization problem, which is based on maximizing classification accuracy to provide the optimal parameters (κ and β), is defined as:

$$\begin{aligned} P\text{: Maximize} \quad & \frac{100}{n}\sum_{r=1}^{n} f_r(\kappa,\beta) \\ \text{Subject to:} \quad & \kappa \in \{2,...,9\}; \\ & \beta \in [0,1] \end{aligned} \tag{12}$$

where the objective or fitness function f depends on the classification accuracy and n represents the set of training objects/samples to be assigned to different classes. The procedure for calculating the fitness function f is described in Algorithm 3. In this regard, the result of the optimization problem defined in Eq. (12) can vary within the interval [0, 100].

Algorithm 3. Procedure to calculate objective function f.

Step 1: Apply k-Means (based on generated κ) to discretize the attributes
Step 2: Build the prototypes based on generated β, according to Algorithm 4
Step 3: Perform the classification procedure according to Algorithm 1
Step 4: Compare the value of the new class with the true class C as follows:
Return the value 1 if object a_r belongs to the class C of a_r, or 0 otherwise

Algorithm 4. Building the classification model for PROAFTN.

Determine a ***threshold*** β as reference for interval selection
$k \leftarrow$ Number of classes
$i \leftarrow$ Prototype's index
$m \leftarrow$ Number of attributes
$\kappa \leftarrow$ Number of intervals/(clusters) for each attribute
$I^r_{jh} \leftarrow$ Intervals $\{S^{1r}_{jh}, S^{2r}_{jh}\}$ for each attribute g_j in each class C^h
$\Re \leftarrow$ Percentage of values within the interval I^r_{jh} per class
Generate PROAFTN intervals according to algorithm 2
$p \leftarrow 0$
for $h \leftarrow 1$, k **do**
  $i \leftarrow 0$
  for $g \leftarrow 1$, m **do**
    for $r \leftarrow 1$, κ **do**
      if $\Re$ of $I^r_{jh} \geq \beta$ **then**
        Choose this interval to be part of the prototype b^h_i
        Go to next attribute g_{m+1}
      else
        Discard this interval and find another one (*i.e.*, I^{r+1}_{jh})
      end if
    end for
  end for
  if $(b^h_i \neq \emptyset \; \forall g_{jh})$ **then** $i \leftarrow i + 1$
  end if
  (Prototypes' composition):
  The selected branches from attribute g_1 to attribute g_m represent the induced prototypes for the class C^h
end for

3.2 Learning PROAFTN Using Particle Swarm Optimization

A new methodology based on the particle swarm optimization (PSO) algorithm was introduced to learn PROAFTN. First, an optimization model was formulated, and thereafter a PSO was used to solve it. PSO was proposed to induce the classification model for PROAFTN in so-called PSOPRO by inferring the best parameters from data with high classification accuracy. It was found that PSOPRO is an efficient approach for data classification. The performance of PSOPRO applied to different classification datasets demonstrates that PSOPRO outperforms the well-known classification methods.

PSO is an efficient evolutionary optimization algorithm using the social behavior of living organisms to explore the search space. Furthermore, PSO is easy to code and requires few control parameters [17]. The proposed approach employs PSO for training and improving the efficiency of the PROAFTN classifier. In this perspective, the optimization model is first formulated, and thereafter a PSO algorithm is used for solving it. During the learning stage, PSO uses training samples to induce the best PROAFTN parameters in the form of prototypes. Then, these prototypes, which represent the classification model, are used for assigning unknown samples. The target is to obtain the set of prototypes that maximizes the classification accuracy on each dataset.

The general description of the PSO methodology and its application is described in [6]. As discussed earlier, to apply PROAFTN, the pessimistic interval $[S^1_{jh}, S^2_{jh}]$ and the optimistic interval $[q^1_{jh}, q^2_{jh}]$ for each attribute in each class need to be determined, where:

$$q^1_{jh} = S^1_{jh} - d^1_{jh} \qquad q^2_{jh} = S^2_{jh} + d^2_{jh} \tag{13}$$

applied to:

$$q^1_{jh} \leq S^1_{jh} \qquad q^2_{jh} \geq S^2_{jh} \tag{14}$$

Hence, $S^1_{jh} = S^1_j(b^h_i)$, $S^2_{jh} = S^2_j(b^h_i)$, $q^1_{jh} = q^1_j(b^h_i)$, $q^2_{jh} = q^2_j(b^h_i)$, $d^1_{jh} = d^1_j(b^h_i)$, and $d^2_{jh} = d^2_j(b^h_i)$.

As mentioned above, to apply PROAFTN, the intervals $[S^1_{jh}, S^2_{jh}]$ and $[q^1_{jh}, q^2_{jh}]$ satisfy the constraints in Eq. (14) and the weights w_{jh} must be obtained for each attribute g_j in class C^h. To simplify the constraints in Eq. (14), the variable substitution based on Eq. (13) is used. As a result, the parameters d^1_{jh} and d^2_{jh} are used instead of q^1_{jh} and q^2_{jh}, respectively. Therefore, the optimization problem, which is based on maximizing classification accuracy providing the optimal parameters $S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}$ and w_{jh}, is defined here,

$$\begin{aligned} P\text{: Maximize} \quad & f(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}, w_{jh}) \\ \text{Subject to:} \quad & S^1_{jh} \leq S^2_{jh}; d^1_{jh}, d^2_{jh} \geq 0 \\ & \sum_{j=1}^{m} w_{jh} = 1 \\ & 0 \leq w_{jh} \leq 1 \end{aligned} \tag{15}$$

where f is the function that calculates the classification accuracy, and n represents the number of training samples used during the optimization. The procedure for calculating the fitness function $f(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}, w_{jh})$ is described in Table 3.

Table 3. The steps for calculating the objective function f.

For all $a \in A$:
Step 1: - Apply the classification procedure according to Algorithm 1
Step 2: - Compare the value of the new class with the true class C^h
- Identify the number of misclassified and unrecognized objects
- Calculate the classification accuracy (*i.e.* the fitness value):
$f = \frac{\text{number of correctly classified objects}}{n}$

To solve the optimization problem presented in Eq. (15), PSO is adopted here. The problem dimension D (*i.e.,* the number of parameters in the optimization problem) is described as follows: Each particle $\mathbf{x}$ is composed of the parameters $S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}$ and w_{jh}, for all $j = 1, 2, ..., m$ and $h = 1, 2, ..., k$. Therefore, each particle in the population is composed of $D = 5 \times m \times k$ real values (*i.e.,* $D = dim(\mathbf{x})$).

3.3 Differential Evolution for Learning PROAFTN

A new learning strategy based on the Differential Evolution (DE) algorithm was proposed for obtaining the best PROAFTN parameters. The proposed strategy is called DEPRO. DE is an efficient metaheuristics optimisation algorithm based on a simple mathematical structure that mimics a complex process of evolution. Based on results generated from a variety of public datasets, DEPRO provides excellent results, outperforming the most common classification algorithms.

In this direction, a new learning approach based on DE is proposed for learning the PROAFTN method. More particularly, DE is introduced here to solve the optimization problem introduced in Eq. (15). The new proposed learning technique, called DEPRO, utilizes DE to train and improve the PROAFTN classifier. In this context, DE is utilized as an inductive learning approach to infer the best PROAFTN parameters from the training samples. The generated parameters are then used to compose the prototypes, which represent the classification model that will be used for assigning unknown samples. The target is to find the prototypes that maximize the classification accuracy on each dataset. The full description of the DE methodology and its application to learn PROAFTN is described in [4]. The general procedure of the DE algorithm is presented in Algorithm 5.

The procedure for calculating the fitness function $f(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}, w_{jh})$ is described in Table 3. The mutation and crossover steps to update the elements (genes) of the trial individual $\mathbf{v}_i$ based DEPRO are performed as follows:

Algorithm 5. Differential Evolution Steps.

Initialization
Evolution
repeat
Mutation
Recombination
Evaluation
Selection
until (termination criteria are met)

$$v_{ihj\tau} = \begin{cases} x_{r_1hj\tau} + F(x_{r_2hj\tau} - x_{r_3hj\tau}), & \text{if } (rand_\tau < \kappa) \text{ or } (\rho = \tau) \\ x_{ihj\tau}, & \text{otherwise.} \end{cases} \tag{16}$$

$$i, r_1, r_2, r_3 \in \{1, ..., N_{pop}\}, \quad i \neq r_1 \neq r_2 \neq r_3;$$
$$h = 1, ..., k; \quad j = 1, ..., m; \quad \tau = 1, ..., D$$

where F is the mutation factor $\in [0, 2]$, and κ is the crossover factor. This modified operation (*i.e.,* Eq. (16)) forces the mutation and crossover process to be applied on each gene τ selected randomly for each set of 5 parameters $S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}$ and w_{jh} in $\mathbf{v}_i$ for all $j = 1, 2, ..., m$ and $h = 1, 2, ..., k$.

3.4 A Hybrid Metaheuristic Framework for Establishing PROAFTN Parameters

As discussed earlier, there are different ways to classify the behavior of metaheuristic algorithms based on their characteristics. One of these major characteristics is to identify whether the evolution strategy is based on population-based search or single point search. Population-based methods deal in every iteration with a set of solutions rather than with a single solution. As a result, population-based algorithms have the capability to efficiently explore the search space, whereas the strength of single-point solution methods is that they provide a structured way to explore a promising region in the search space. Therefore, a promising area in the search space is searched in a more intensive way by using single-point solution methods than by using population-based methods [58]. Population-based methods can be augmented with single-point solution methods to improve the search mechanism. While the use of population-based methods ensures an exploration of the search space, the use of single-point techniques helps to identify good areas in the search space. One of the most popular ways of hybridization concerns the use of single-point search methods in population-based methods. Thus, hybridization that in some way manages to combine the advantages of population-based methods with the strengths of single-point methods is often very successful, which is the motivation and the case for this work. In many applications, hybrids metaheuristics have proved to be quite beneficial in improving the fitness of individuals [37,38,57]. In this methodology, a new hybrid of metaheuristics approaches were introduced to

obtain the best PROAFTN parameters configuration for a given problem. The two proposed hybrid approaches are: (1) Particle Swarm optimization (PSO) and Reduced Variable Neighborhood Search (RVNS), called PSOPRO-RVNS; and (2) Differential Evolution (DE) and RVNS, called DEPRO-RVNS. Based on the generated results on both training and testing data, it was shown that the performance of PROAFTN is significantly improved compared with the previous study presented in the previous sections (Sects. 3.2 and 3.3). Furthermore, the experimental study demonstrated that PSOPRO-RVNS and DEPRO-RVNS strongly outperform well-known machine learning classifiers in a variety of problems. RVNS is a variation of the metaheuristic Variable Neighborhood Search (VNS) [33,34]. The basic idea of the VNS algorithm is to find a solution in the search space with a systematic change of neighborhood. The basic VNS is very useful for approximate solutions for many combinatorial and global optimization problems; however, the major limitation is that it is very time consuming because of the utilization of ingredient-based approaches as it is used as a local search routine. RVNS uses a different approach; the solutions are drawn randomly from their neighborhood. The incumbent solution is replaced if a better solution is found. RVNS is simple, efficient and provides good results with low computational cost [30,34]. In RVNS, two procedures are used: shake and move. Starting from the initial solution (the position of prematurely converged individuals) $\mathbf{x}$, the algorithm selects a random solution $\mathbf{x}'$ from the initial solution's neighborhood. If the generated $\mathbf{x}'$ is better than $\mathbf{x}$, it replaces $\mathbf{x}$ and the algorithm starts all over again with the same neighborhood. Otherwise, the algorithm continues with the next neighborhood structure. The pseudo-code of RVNS is given in Algorithm 6.

Algorithm 6. Random Variable Neighborhood Search steps.

Require:
Define neighborhood structures N_k for $k = 1, 2, \ldots, k_{max}$, that will be used in the search
Get the initial solution $\mathbf{x}$ and choose stopping condition
$k \leftarrow 1$
while $k < k_{kmax}$ **do**
  Shaking:
    Generate a point $\mathbf{x}'$ at random from the k-th neighborhood of $\mathbf{x}$ ($\mathbf{x}' \in N_k(\mathbf{x})$)
  Move or not:
  if $\mathbf{x}'$ is better than the incumbent $\mathbf{x}$ **then**
    $\mathbf{x} \leftarrow \mathbf{x}'$
    $k \leftarrow 1$
  else
    set $k \leftarrow k + 1$
  end if
end while

In [13] the RVNS heuristics is used to learn the PROAFTN classifier by optimizing its parameters that are presented as intervals namely the pessimistic and

optimistic intervals. In this light, a hybrid of metaheuristics is proposed here for training the PROAFTN method. In this regard, the two different hybrid approaches PSO augmented with RVNS (called PSOPRO-RVNS) and DE augmented with RVNS (called DEPRO-RVNS) are proposed for solving this optimization problem. The two proposed training techniques presented in (Sects. 3.2 and 3.3) are integrated with the single point search RVNS, to improve the performance of PROAFTN. The details on how DE and RVNS have been used together to learn the PROAFTN classifier is described in [5]. And in the same context, the details of the application of PSO and RVNS to learn PROAFTN is described in [3]. To use RVNS to find a better solution provided by PSO or DE in each iteration, the following equations are considered to update the boundary for the previous solution $\mathbf{x}$ containing $(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh})$ parameters:

$$l_{\lambda jbh} = x_{\lambda jbh} - (k/k_{max})x_{\lambda jbh} \tag{17}$$

$$use\, x instead\, of\, su_{\lambda jbh} = x_{\lambda jbh} + (k/k_{max})x_{\lambda jbh} \tag{18}$$

where $l_{\lambda jbh}$ and $u_{\lambda jbh}$ are the lower and upper bounds for each element $\lambda \in [1, \ldots, D]$. Factor k/k_{max} is used to define the boundary for each element and $x_{\lambda jbh}$ is the previous solution for each element $\lambda \in [1, \ldots, D]$ provided by PSO.

The use of the hybrid PSO/DE augmented with RVNS for learning PROAFTN is explained here and for more details please see [5]. Using PSO, the elements for each particle position $\mathbf{x}_i$ consisting of the parameters $S^1_{jh}, S^2_{jh}, d^1_{jh}$ and d^2_{jh} are updated using:

$$x_{i\lambda jbh}(t+1) = x_{i\lambda jbh}(t) + v_{i\lambda jbh}(t+1) \tag{19}$$

where the velocity update $\mathbf{v}_i$ for each element based on $\mathbf{P}_i^{Best}$ and $\mathbf{G}^{Best}$ is formulated as:

$$\begin{aligned} v_{i\lambda jbh}(t+1) = \varpi(t) v_{i\lambda jbh}(t) + \\ \tau_1 \rho_1 (P^{Best}_{i\lambda jbh} - x_{i\lambda jbh}(t)) + \\ \tau_2 \rho_2 (G^{Best}_{\lambda jbh} - x_{i\lambda jbh}(t)) \end{aligned} \tag{20}$$

$$i = 1, ..., N_{pop}; \quad \lambda = 1, ..., D$$

$$j = 1, ..., m; \quad b = 1, ..., L_h; \quad h = 1, ..., k$$

where $\varpi(t)$ is the inertia weight that controls the exploration of the search space. τ_1 and τ_2 are the individual and social components/weights, respectively. ρ_1 and ρ_2 are random numbers between 0 and 1. $\mathbf{P}_i^{Best}(t)$ is the personal best position of the particle i, and $\mathbf{G}^{Best}(t)$ is the neighborhood best position of particle i. Algorithm 6 demonstrates the required steps to evolve the velocity $\mathbf{v}_i$ and particle position $\mathbf{x}_i$ for each particle containing PROAFTN parameters. The *shaking* phase to randomly generate the elements of $\mathbf{x}'$ is given by:

$$x'_{\lambda jbh} = l_{\lambda jbh} + (u_{\lambda jbh} - l_{\lambda jbh}).rand[0, 1] \tag{21}$$

Accordingly, the *moving* is applied as:

$$\text{If } f'(x'_{\lambda jbh}) > f(x_{\lambda jbh}) \text{ then } x_{\lambda jbh} = x'_{\lambda jbh} \tag{22}$$

The steps that explain the employment of RVNS to improve PROAFTN parameters are listed in Algorithm 7.

Algorithm 7. The RVNS heuristic for learning the classification method PROAFTN

Require:
Get PSO or DE premature-solution as initial solution $\mathbf{x}$ which contains $S^1_{jh}, S^2_{jh}, d^1_{jh}$ and d^2_{jh}
Calculate the objective function $f(\mathbf{x})$ of the optimization problem in Eq. (15).
Stopping condition k is set to 4
repeat
$k \leftarrow 1$
Shaking:
while $k < k_{kmax}$ **do**
for each parameter of parameters $(S^1_{jh}, S^2_{jh}, d^1_{jh}, d^2_{jh}) \in \mathbf{x}$ **do**
Update the boundary for each parameter according to Eqs. (17 and 18)
Randomly generate new position $\mathbf{x}'$ from k-th neighborhood for $\lambda^{th} \in N_k(\tau)$ (Eq. (21))
end for
Submit $\mathbf{x}'$ to calculate the new fitness value (f') according to Eq. (15)
Move or not (Eq. (22)):
if $f'(x')$ is better than the incumbent $f(\mathbf{x})$ **then**
$\mathbf{x} \leftarrow \mathbf{x}'$
$k \leftarrow 1$
else
set $k \leftarrow k + 1$
end if
end while
until stopping condition is met
return the best generated point $\mathbf{x}'$ to PSO or DE to continue the search

4 Comparative Study with PROAFTN and Well Known Classifiers

The proposed methodologies were implemented in Java and applied to 12 popular datasets: Breast Cancer Wisconsin Original (BCancer), Transfusion Service Center (Blood), Heart Disease (Heart), Hepatitis, Haberman's Survival (HM), Iris, Liver Disorders (Liver), Mammographic Mass (MM), Pima Indians Diabetes (Pima), Statlog Australian Credit Approval (STAust), Teaching Assistant Evaluation (TA), and Wine. The details of the datasets' description and their dimensionality are presented in Table 4. The datasets are in the public domain and are available at the University of California at Irvine (UCI) Machine Learning Repository database [8].

Table 4. Description of datasets used in our experiments.

	Dataset	Instances	Attributes	Classes
1	BCancer	699	9	2
2	Blood	748	4	2
3	Heart	270	13	2
4	Hepatitis	155	19	2
5	HM	306	3	2
6	Iris	150	4	3
7	Liver	345	6	2
8	MM	961	5	2
9	Pima	768	8	2
10	STAust	690	14	2
11	TA	151	5	3
12	Wine	178	13	3

To summarize, a comparison of the various approaches introduced throughout this research for learning PROAFTN – GAPRO, PSOPRO, DEPRO, PSOPRO-RVNS and DEPRO-RVNS – is presented in Table 5. One can see that DEPRO-RVNS and PSOPRO-RVNS perform the best.

Table 5. The performance of all approaches for learning PROAFTN introduced in this research study based on classification accuracy (in %). The average accuracy and average ranking is also included.

Dataset	GA-PRO	PSOPRO	DEPRO	PSOPRO-RVNS	DEPRO-RVNS
BCancer	96.76	97.14	96.97	97.33	97.05
Blood	75.43	79.25	79.59	79.46	79.61
HM	83.85	84.27	83.74	84.36	83.81
Heart	71.95	86.04	84.17	87.05	85.37
Hepatitis	73.84	75.73	80.36	76.27	76.10
Iris	96.57	96.21	96.47	96.30	96.66
Liver	71.83	69.31	71.01	70.97	70.99
MM	84.92	82.31	84.33	84.07	84.77
Pima	72.19	77.47	75.37	77.42	77.23
STAust	81.78	86.09	85.62	86.10	86.04
TA	52.44	60.55	61.80	60.62	62.72
Wine	97.33	96.79	96.87	96.72	97.10
Average accuracy	79.91	82.60	83.03	83.06	**83.12**
Average rank	3.58	3.33	3.08	2.58	**2.42**

Table 7 summarizes and gives robust analysis on a comparison that includes the developed approaches of learning PROAFTN classifier against other classifiers. As observed, both approaches DEPRO-RVNS and PSOPRO-RVNS strongly outperform other classifiers. Therefore, the developed approaches can be classified into three groups, based on their performances:

- Best approaches: DEPRO-RVNS and PSOPRO-RVNS.
- Middle approaches: DEPRO and PSOPRO.
- Weakest approach: GA-PRO.

It should be noted also that DEPRO-RVNS and PSOPRO-RVNS are efficient in terms of computation speed. One of the advantages of DE and PSO over other global optimization methods is that they often converge faster and with more certainty than other methods. Furthermore, utilizing RVNS inside DE and PSO improved the search for good solutions in a shorter time (Table 5).

Table 6. Experimental results based on classification accuracy (in %) to measure the performance of the well-known classifiers on the same datasets

Dataset	C4.5 J48	NB	SVM SMO	NN MLP	k-NN Ibk, k = 3	PART	RForest n = 500	GLM	Deep learning
BCancer	94.56	95.99	96.70	95.56	97.00	97.05	97.4	97.9	97.9
Blood	77.81	75.40	76.20	78.74	74.60	79.61	76.1	74.9	78.7
Heart	76.60	83.70	84.10	78.10	78.89	73.33	57.6	60.4	54.9
Hepatitis	80.00	85.81	83.87	81.94	84.52	82.58	90.1	92.6	94.8
HM	71.90	74.83	73.52	72.87	70.26	72.55	73.1	69.2	67.2
Iris	96.00	96.00	96.00	97.33	95.33	94.00	95.3	96.7	90.7
Liver	68.70	56.52	58.26	71.59	61.74	63.77	71.8	73.0	74.1
MM	82.10	78.35	79.24	82.10	77.21	82.21	80.8	84.9	84.7
Pima	71.48	75.78	77.08	75.39	73.44	73.05	77.4	78.3	75.4
STAust	85.22	77.25	85.51	84.93	83.62	83.62	86.7	88.9	86.8
TA	59.60	52.98	54.30	54.30	50.33	58.28	66.1	52.3	39.6
Wine	91.55	97.40	99.35	97.40	95.45	92.86	97.8	98.9	97.7

Comparison with was done against implementations provided in WEKA [27] for neural network multi-level perceptron (NN MLD), naive Bayes (NB), decision trees (PART), C4.5 and k nearest neighbour (knn). We used H2O for deep learning (h2o DL) [19] and generalized linear models (h2o GLM) [44]. We used R's implementation of random forest (RFOREST) [41] with n = 500 trees. PROAFTN and decision trees share a very important property: both of them use the white box model. Decision trees and PROAFTN can generate classification models which can be easily explained and interpreted. However, when evaluating any classification method there is another important factor to be considered:

Table 7. Mean accuracy rankings. The algorithms developed in this paper are marked in bold.

Algorithm	Mean rank
DEPRO-RVNS	4.75
PSOPRO-RVNS	4.75
h2o GLM	5.29
PSOPRO	5.50
DEPRO	6.08
RForest 500	6.25
h2o DL	7.04
GA-PRO	8.08
SVM SMO	8.12
NN MLP	8.12
NB	9.54
PART	9.62
C4.5	10.62
k-NN	11.21

classification accuracy. Based on the experimental study presented in Sect. 4, the PROAFTN method has proven to generate a higher classification accuracy than decision tree such as C4.5 [46] and other well-known classifiers learning algorithms including Naive Bayes, Support Vector Machines (SVM), Neural Network (NN), K- Nearest Neighbor K-NN, and Rule Learner (see Table 6). That can be explain by the fact that PROAFTN using fuzzy intervals. A general comparison between PROAFTN based on the proposed learning approaches adopted in this paper (PRO-BPLA) and other machine learning classifiers is summarized in Table 8. The observations made in this table are based on evidence of existing empirical and theoretical studies as presented in [40]. We have also added some evidence based on the results obtained using the developed learning methodology introduced in this research study. As a summary, Table 8 compares the properties of some well known machine learning classifiers against the properties of the classification method PROAFTN.

In this chapter, we have presented the implementation of machine learning and metaheuristics algorithms for parameters training of multicriteria classification method. We have shown that learning techniques based on metaheuristics proved to be a successful approach for optimizing the learning of PROAFTN classification method and thus greatly improving its performances. As has been demonstrated, every classification algorithm has its strengths and limitations. More particularly, the characteristics of the method and whether it is strong or weak depend on the situation or on the problem. For instance, assume the problem at hand is a medical dataset and the interest is to look for a classification method for medical diagnostics. Suppose the executives and experts are looking

Table 8. Summary of the of well-known classifiers versus PRO-BPLA properties (the best rating is **** and the worst is *)

	DT	NB	SVM	NN	*k*-NN	PART	PRO-BPLA	RForest	GLM	Deep Learning
Accuracy in general	**	*	***	***	**	**	****	****	****	****
Dealing with discrete/-continuous attributes	***not directly	** not continuous	** not discrete	** not discrete	** not directly discrete	** not directly continuous	**** continuous & discrete	** not directly continuous	** not directly discrete	** not directly discrete
Tolerance to noise	**	**	**	*	*	*	***	**	**	***
Training time	**	***	*	*	****	**	*	**	*	*
Testing time	****	***	****	****	*	****	***	***	****	****
Dealing with danger of overfitting	**	***	**	*	***	**	**	***	*	*
Model parameter handling	***	***	*	*	***	***	****	****	**	*
Interpretability	****	****	*	*	**	****	****	***	*	*

for a high level of classification accuracy and at the same time they are very keen to know more details about the classification process (*e.g.,* why the patient is classified to this category of disease). In such circumstances, classifiers such as Deep Learning networks, *k*-NN, or SVM may not be an appropriate choice, because of the limited interpret-ability of their classification models. Although deep learning networks have been successfully applied to some health-care application and in particularly into medical imaging, they suffered from some limitations such as the limited interpret-ability of their classification results; they require a very large balanced labeled data set; the preprocessing or change of input domain is often required to bring all the input data to the same scale [48]. Thus, there is a need to look for other classifiers that reason about their outputs and can generate good classification accuracy, such as DTs (C4.5, ID3), NB, or PROAFTN.

Based on the experimental and the comparative study presented in Table 8, the PROAFTN method based on our proposed learning approaches has good accuracy in most instances and can deal with all types of data without sensitivity to noise. PROAFTN uses the pairwise comparison and therefore, there is no need for looking for suitable normalization technique of data like the case of other classifiers. Furthermore, PROAFTN is a transparent and interpretable classifier where it's easy to generalize the classification rules from the obtained prototypes. It can use both approaches deductive and inductive learning, which allow us to use in the same time historical data with expert judgment to compose the classi-

fication model. To sum up, there is no complete or comprehensive classification algorithm that can handle or fit all classification problems. In response to this deficiency, the major task of this work is to review an integration of methodologies from three major fields, MCDA, machine learning, and optimization based metaheuristics, through the aforementioned classification method PROAFTN. The target of this study was to exploit the machine learning techniques and the optimization approaches to improve the performance of PROAFTN. The aim is to find a good suitable and comprehensive (interpretable) classification procedure that can be applied efficiently in many applications including the ambient assisted living environments.

5 Conclusions and Future Work

The target of this chapter is to exploit the machine learning techniques and the optimization approaches to improve the performance of PROAFTN. The aim is to find a good suitable and comprehensive (interpretable) classification procedure that can be applied efficiently in health applications including the ambient assisted living environments. This chapter describes the ability of the metaheuristics when embedded to the classification method PROAFTN in order to classify new objects. To do this we compared the improved PROAFTN methodology with those reported previously on the same data and same validation technique (10-cross validation). In addition to reviewing several approaches to modeling and learning classification method PROAFTN, this chapter also presents new ideas to further research in the areas of data mining and machine learning. Below are some possible directions for future research.

1. The fact that PROAFTN has several parameters to be obtained for each attribute and for each class, which provides more information to assign objects to the closest class. However, in some cases this may cause some limitation on the speed of learning, particularly when using metaheuristics, as we presented in this paper. Possible future solutions could be summarized as follows:
   - Utilizing different approaches for obtaining the weights. One possible direction is to use a features ranking approach by using some strong algorithms that perform well in the aspect of dimensionality reduction.
   - Determining intervals bounds for more than one prototype before performing optimization. This would involve establishing the intervals' bounds *a priori* by using some clustering techniques, hence improving and speeding up the search and improving the likelihood of finding the best solutions.
2. As we know the performance of approaches based on the choice of control parameters varies from one application to another. However, in this work the control parameters are fixed for all applications. A better control of parameter choice for the metaheuristics based PROAFTN algorithms will be investigated.

3. To speed up the PROAFTN learning process, possible improvement could be made by using parallel computation. The different processors can deal with the fold independently in the cross validation folds process. The parallelism can be also applied in the composition of prototypes of each class.
4. In this chapter, an inductive learning is presented to build the classification models for the PROAFTN method. PROAFTN also can apply the deductive learning that allows the introduction of the given knowledge in setting PROAFTN parameters such intervals and/or weights to build the prototype of classes.

References

1. Al-Obeidat, F., Al-Taani, A.T., Belacel, N., Feltrin, L., Banerjee, N.: A fuzzy decision tree for processing satellite images and landsat data. Procedia Comput. Sci. **52**, 1192–1197 (2015)
2. Al-Obeidat, F., Belacel, N.: Alternative approach for learning and improving the MCDA method PROAFTN. Int. J. Intell. Syst. **26**(5), 444–463 (2011)
3. Al-Obeidat, F., Belacel, N., Carretero, J.A., Mahanti, P.: Automatic parameter settings for the PROAFTN classifier using hybrid particle swarm optimization. In: Farzindar, A., Kešelj, V. (eds.) AI 2010. LNCS (LNAI), vol. 6085, pp. 184–195. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13059-5_19
4. Al-Obeidat, F., Belacel, N., Carretero, J.A., Mahanti, P.: Differential evolution for learning the classification method PROAFTN. Knowl.-Based Syst. **23**(5), 418–426 (2010)
5. Al-Obeidat, F., Belacel, N., Carretero, J.A., Mahanti, P.: A hybrid metaheuristic framework for evolving the PROAFTN classifier. Spec. J. Issues World Acad. Sci. Eng. Technol. **64**, 217–225 (2010)
6. Al-Obeidat, F., Belacel, N., Carretero, J.A., Mahanti, P.: An evolutionary framework using particle swarm optimization for classification method PROAFTN. Appl. Soft Comput. **11**(8), 4971–4980 (2011)
7. Al-Obeidat, F., Belacel, N., Mahanti, P., Carretero, J., et al.: Discretization techniques and genetic algorithm for learning the classification method PROAFTN. In: International Conference on Machine Learning and Applications, ICMLA 2009, pp. 685–688. IEEE (2009)
8. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
9. Ban, A., Coroianu, L.: Simplifying the search for effective ranking of fuzzy numbers. IEEE Trans. Fuzzy Syst. **23**(2), 327–339 (2015). https://doi.org/10.1109/TFUZZ.2014.2312204
10. Belacel, N.: Multicriteria classification methods: methodology and medical applications. Ph.D. thesis, Free University of Brussels, Belgium (1999)
11. Belacel, N.: Multicriteria assignment method PROAFTN: methodology and medical application. Eur. J. Oper. Res. **125**(1), 175–183 (2000)
12. Belacel, N., Boulassel, M.: Multicriteria fuzzy assignment method: a useful tool to assist medical diagnosis. Artif. Intell. Med. **21**(1–3), 201–207 (2001)
13. Belacel, N., Raval, H., Punnen, A.: Learning multicriteria fuzzy classification method PROAFTN from data. Comput. Oper. Res. **34**(7), 1885–1898 (2007)
14. Belacel, N., Vincke, P., Scheiff, J., Boulassel, M.: Acute leukemia diagnosis aid using multicriteria fuzzy assignment methodology. Comput. Methods Programs Biomed. **64**(2), 145–151 (2001). https://doi.org/10.1016/S0169-2607(00)00100-0

15. Belacel, N., Wang, Q., Richard, R.: Web-integration of PROAFTN methodology for acute leukemia diagnosis. Telemed. J. e-Health **11**(6), 652–659 (2005)
16. Belacel, N., Al-Obeidat, F.: A learning method for developing PROAFTN classifiers and a comparative study with decision trees. In: Butz, C., Lingras, P. (eds.) AI 2011. LNCS (LNAI), vol. 6657, pp. 56–61. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21043-3_7
17. van den Bergh, F., Engelbrecht, A.: A study of particle swarm optimization particle trajectories. Inf. Sci. **176**(8), 937–971 (2006). https://doi.org/10.1016/j.ins.2005.02.003
18. Brasil Filho, A.T., Pinheiro, P.R., Coelho, A.L.V., Costa, N.C.: Comparison of two MCDA classification methods over the diagnosis of Alzheimer's disease. In: Wen, P., Li, Y., Polkowski, L., Yao, Y., Tsumoto, S., Wang, G. (eds.) RSKT 2009. LNCS (LNAI), vol. 5589, pp. 334–341. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02962-2_42
19. Candel, A., Parmar, V., LeDell, E., Arora, A., Lanford, J.: Deep Learning with H2O, September 2016. http://h2o.ai/resources
20. Ching, J., Wong, A.K., Chan, K.: Class-dependent discretization for inductive learning from continuous and mixed-mode data. IEEE Trans. Pattern Anal. Mach. Intell. **17**(7), 641–651 (1995)
21. Crammer, K., Singer, Y.: On the learnability and design of output codes for multiclass problems. Mach. Learn. **47**(2–3), 201–233 (2002)
22. Doumpos, M., Zopounidis, C.: A multicriteria classification approach based on pairwise comparisons. Eur. J. Oper. Res. **158**(2), 378–389 (2004)
23. Dubois, D., Prade, H., Sabbadin, R.: Decision theoritic foundations of qualitative possibility theory. Eur. J. Oper. Res. **128**, 459–478 (2015)
24. El-Alfy, E.S.M., Al-Obeidat, F.N.: A multicriterion fuzzy classification method with greedy attribute selection for anomaly-based intrusion detection. Procedia Comput. Sci. **34**, 55–62 (2014)
25. El-Alfy, E.S.M., Al-Obeidat, F.N.: Detecting cyber-attacks on wireless mobile networks using multicriterion fuzzy classifier with genetic attribute selection. Mob. Inf. Syst. **501**, 585432 (2015)
26. Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. In: XIII International Joint Conference on Artificial Intelligence (IJCAI 1993), pp. 1022–1029 (1993)
27. Frank, E., Hall, M.A., Witten, I.H.: The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Fourth edn. Morgan Kaufmann, Burlington (2016)
28. Garcia, S., Luengo, J., Saez, V., Herrera, F.: A survey of discretization techniques: taxonomy and empirical analysis in supervised learning. IEEE Trans. Knowl. Data Eng. **25**(4), 734–750 (2013). https://doi.org/10.1109/TKDE.2012.35
29. García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J.M., Herrera, F.: Data discretization: taxonomy and big data challenge. WIREs Data Mining Knowl. Discov. **6**, 5–21 (2016). https://doi.org/10.1002/widm.1173
30. Glover, F.W., Kochenberger, G.A.: Handbook of Metaheuristics. Kluwer Academic Publishers, Norwell (2003)
31. Goebel, M., Gruenwald, L.: A survey of data mining and knowledge discovery software tools. ACM SIGKDD Explor. Newslett. **1**(1), 20–33 (1999)
32. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional, Boston (1989)
33. Hansen, P., Mladenovic, N.: Variable neighborhood search for the p-median. Location Sci. **5**(4), 207–226 (1997)

34. Hansen, P., Mladenovic, N.: Variable neighborhood search: principles and applications. Eur. J. Oper. Res. **130**(3), 449–467 (2001)
35. Ishizaka, A., Nemery, P.: Assigning machines to incomparable maintenance strategies with electre-sort. Omega **47**, 45–59 (2014). https://doi.org/10.1016/j.omega.2014.03.006
36. Ivascu, T., Cincar, K., Dinis, A., Negru, V.: Activities of daily living and falls recognition and classification from the wearable sensors data. In: E-Health and Bioengineering Conference (EHB), pp. 627–630. IEEE (2017)
37. Jung, S., Moon, B.: A hybrid genetic algorithm for the vehicle routing problem with time windows. In: GECCO, pp. 1309–1316 (2002)
38. Kim, J.P., Moon, B.R.: A hybrid genetic search for circuit bipartitioning. In: GECCO, p. 685 (2002)
39. Kotsiantis, S.: Supervised machine learning: a review of classification techniques. Informatica **31**, 249–268 (2007)
40. Kotsiantis, S.B., Zaharakis, I.D., Pintelas, P.E.: Machine learning: a review of classification and combining techniques. Artif. Intell. Rev. **26**(3), 159–190 (2006)
41. Law, A.: Breiman and Cutler's Random Forests for Classification and Regression, October 2015. https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
42. Marchant, T.: A measurement-theoretic axiomatization of trapezoidal membership functions. IEEE Trans. Fuzzy Syst. **15**(2), 238–242 (2007). https://doi.org/10.1109/TFUZZ.2006.880000
43. Monekosso, D., Florez-Revuelta, F., Remagnino, P.: Ambient assisted living [guest editors' introduction]. IEEE Intell. Syst. **30**(4), 2–6 (2015). https://doi.org/10.1109/MIS.2015.63
44. Nykodym, T., Kraljevic, T., Hussami, N., Rao, A., Wang, A.: Generalized Linear Models with H2O, September 2016. http://h2o.ai/resources
45. Perny, P., Roy, B.: The use of fuzzy outranking relations in preference modelling. Fuzzy Sets Syst. **49**, 33–53 (1992)
46. Quinlan, J.R.: Improved use of continuous attributes in C4.5. J. Artif. Intell. Res. **4**, 77–90 (1996)
47. Ranasinghe, S., Machot, F.A., Mayr, H.C.: A review on applications of activity recognition systems with regard to performance and evaluation. Int. J. Distrib. Sens. Netw. **12**(8) (2016). https://doi.org/10.1177/1550147716665520
48. Rav, D., et al.: Deep learning for health informatics. IEEE J. Biomed. Health Inform. **21**(1), 4–21 (2017). https://doi.org/10.1109/JBHI.2016.2636665
49. Reeves, C.R., Rowe, J.E.: Genetic Algorithms: Principles and Perspectives. A Guide to GA Theory. Kluwer Academic Publishers, Norwell (2002)
50. Resende Monteiro, A.L., Manso Correa Machado, A., Lewer, M., Henrique, M.: A multicriteria method for cervical tumor segmentation in positron emission tomography. In: 2014 IEEE 27th International Symposium on Computer-Based Medical Systems (CBMS), pp. 205–208. IEEE (2014)
51. Roy, B.: Multicriteria Methodology for Decision Aiding. Kluwer Academic, Norwell (1996)
52. Roy, B.: Multicriteria Methodology for Decision Aiding. Nonconvex Optimization and Its Applications. Springer, Heidelberg (2013). https://doi.org/10.1007/978-1-4757-2500-1
53. Sassi, I., Belacel, N., Bouslimani, Y.: Photonic-crystal fibre modeling using fuzzy classification approach. Int. J. Recent Trends Eng. Technol. **6**(2), 100–104 (2011)
54. Sharlig, A.: Décider sur plusieurs critères, panorama de laide à la décision multicritère. Press polytechniques Romandes, Lausanne (1985)

55. Singh, N., Arora, H.: Network intrusion detection using feature selection and PROAFTN classification. Int. J. Sci. Eng. Res. **6**(4), 466–472 (2015)
56. Sobrado, F., Pikatza, J., Larburu, I., Garcia, J., de Ipiña, D.: Towards a clinical practice guideline implementation for asthma treatment. In: Conejo, R., Urretavizcaya, M., Pérez-de-la Cruz, J. (eds.) CAEPIA-TTIA 2003. LNCS, pp. 587–596. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25945-9_58
57. Talbi, E.-G., Rahoual, M., Mabed, M.H., Dhaenens, C.: A hybrid evolutionary approach for multicriteria optimization problems: application to the flow shop. In: Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D. (eds.) EMO 2001. LNCS, vol. 1993, pp. 416–428. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44719-9_29
58. Talbi, E.G.: A taxonomy of hybrid metaheuristics. J. Heuristics **8**(5), 541–564 (2002). https://doi.org/10.1023/A:1016540724870
59. Vincke, P.: Multicriteria Decision-Aid. Wiley, Hoboken (1992). https://books.google.ca/books?id=H2NRAAAAMAAJ
60. Witten, H.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco (2005)
61. Wu, X.: Fuzzy interpretation of discretized intervals. IEEE Trans. Fuzzy Syst. **7**(6), 753–759 (1999)
62. Zopounidis, C., Doumpos, M.: Multicriteria preference disaggregation for classification problems with an application to global investing risk. Decis. Sci. **32**(2), 333–385 (2001)
63. Zopounidis, C., Doumpos, M.: Multicriteria classification and sorting methods: a literature review. Eur. J. Oper. Res. **138**(2), 229–246 (2002)