International Journal of Machine Learning and Cybernetics (2023) 14:2137–2154
https://doi.org/10.1007/s13042-022-01751-z

ORIGINAL ARTICLE

A novel framework based on the multi-label classification for dynamic selection of classifiers

Javad Elmi[1] · Mahdi Eftekhari[1] · Adel Mehrpooya[1,2] · Mohammad Rezaei Ravari[1]

Received: 25 February 2021 / Accepted: 13 December 2022 / Published online: 2 January 2023

Abstract

Multi-classifier systems (MCSs) are some kind of predictive models that classify instances by combining the output of an ensemble of classifiers given in a pool. With the aim of enhancing the performance of MCSs, dynamic selection (DS) techniques have been introduced and applied to MCSs. Dealing with each test sample classification, DS methods seek to perform the task of classifier selection so that only the most competent classifiers are selected. The principal subject regarding DS techniques is how the competence of classifiers corresponding to every new test sample classification task can be estimated. In traditional dynamic selection methods, for classifying an unknown test sample x, first, a local region of data that is similar to x is detected. Then, those classifiers that efficiently classify the data in the local region are also selected so as to perform the classification task for x. Therefore, the main effort of these methods is focused on one of the two following tasks: (i) to provide a measure for identifying a local region, or (ii) to provide a criterion for measuring the efficiency of classifiers in the local region (competence measure). This paper proposes a new version of dynamic selection techniques that does not follow the aforementioned approach. Our proposed method uses a multi-label classifier in the training phase to determine the appropriate set of classifiers directly (without applying any criterion such as a competence measure). In the generalization phase, the suggested method is employed efficiently so as to predict the appropriate set of classifiers for classifying the test sample x. It is remarkable that the suggested multi-label-based framework is the first method that uses multi-label classification concepts for dynamic classifier selection. Unlike the existing meta-learning methods for dynamic ensemble selection in the literature, our proposed method is very simple to implement and does not need meta-features. As the experimental results indicate, the suggested technique produces a good performance in terms of both classification accuracy and simplicity which is fairly comparable with that of the benchmark DS techniques. The results of conducting the Quade non-parametric statistical test corroborate the clear dominance of the proposed method over the other benchmark methods.

Keywords Multi-classifier systems (MCSs) · Dynamic selection (DS) · Competence measure · Multi-label classifiers

1 Introduction

Over the past decades, Multiple Classifier Systems (MCSs) that are essential tools to handle the task of combining diverse classifiers have been assessed and employed as powerful and applicable techniques to deal with challenges raised in diverse classification subjects [42]. In [18], an extensive and in-depth review of MCS-based methods was carried out in which quite a large number of ensemble learning techniques were closely and carefully examined in terms of their synthesis, variety, and dynamic updates. So far, by applying these systems, a large number of practical issues have been addressed [8], such as problems in face recognition [37], anomaly detection [21, 22], credit scoring [3], speech recognition [45, 47], recommender system [35, 36],

✉ Mahdi Eftekhari
m.eftekhari@uk.ac.ir

Javad Elmi
elmi@eng.uk.ac.ir

Adel Mehrpooya
adel.mehrpooya@hdr.qut.edu.au; amehrpooya@eng.uk.ac.ir

Mohammad Rezaei Ravari
mohammad.rezaei@eng.uk.ac.ir

[1] Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman, Iran

[2] School of Mathematical Sciences, Faculty of Science, Queensland university of Technology, Brisbane, Australia

software bug prediction [1, 29], intrusion detection [2, 25, 32] and remote sensing [14, 27, 31] as well as having been successfully used to tackle problems on changing environments [24]. In very recent years, new applications of MCSs have been explored regarding imbalanced data problems [6, 16, 19, 43] and related biological datasets to handle disease detection problems such as COVID-19 diagnosis [10].

A greater number of DS methods are established based on the standard MCS mechanism. As shown in Fig. 1, the standard MCSs are composed of two major phases including training and generalization phases. In the training phase, a pool of classifiers is generated during the Generation step. The generalization phase consists of four steps as follows: local region definition (LR definition), competence level calculation (CL calculation), Selection, and Prediction. First, the local region is defined, and following this, the competence level of classifiers in the local region is computed. Next, the classifiers with the high competence level are selected. Finally, a label is assigned to the input data based on the label predicted by the selected classifiers in the Selection step. Some DS methods focus on enhancing the performance corresponding to the standard steps of MCS mechanism. Overall Local Accuracy (OLA) [39], Dynamic Ensemble Selection Performance (DES-P) [39], K-Nearest Oracles Union (KNORA-U) and K-Nearest Oracles Eliminate (KNORA-E) [23] are notable techniques that fall into this category. However, some other dynamic selection techniques develop a deeper and more precise mechanism by adding new steps to Multiple-Classifier Behavior (MCB). As such, Dynamic ensemble selection-based metadata (META-DES) [9] and Dynamic ensemble selection based on hesitant (DES-Hesitant) [13] are typical benchmark examples of such methods. A brief description of these techniques is presented in the Sect. 2.

The major focus of this paper is to investigate a general rule on which a selector is rested. In addition, a study of the customized rule of the most popular DS techniques is also included. A question then arises, "among all the possible rules, which one is the most appropriate to be applied to selectors"? This is the main question this paper tries to address. Apparently, it is a particularly complex issue to determine the proper rule for selectors. It is worth noting that many studies and researches have been conducted on DS techniques to resolve this problem and in fact, each of the previously presented rules is set through trial and error or by performing experiments on different datasets.

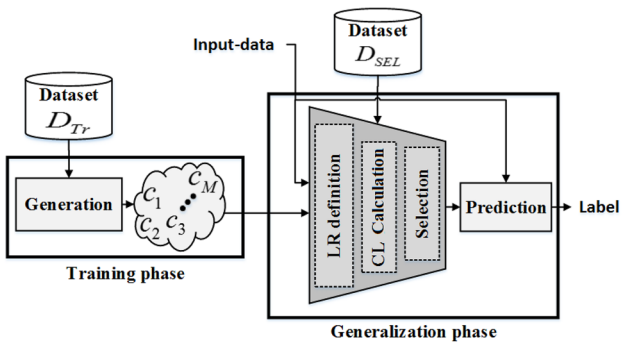

Fig. 1 The main stages of a typical sector in a multiple-classifier system

This paper offers a novel perspective on the aforementioned question that is not based on conducting search operations and enacting strict rules. Moreover, it proposes that no rule should be considered the most proper one for selectors. In specific, the following procedure is set out in this paper to achieve an effective and practical solution to the aforesaid question:

- the selector is a black box whose output is a list of appropriate classifiers for each input data x;
- theoretically, a classifier c is appropriate for the datapoint x if it can classify x correctly;
- considering the previous item, a list of appropriate classifiers for each training data is available;
- therefore, the mentioned black box is a "classifier" and its output is specified on the training data;
- specifically, it is apparent that many DES methods obey the general Rule 1.

Rule 1: "If a classifier c is efficient in classifying the data positioned in a local region (i.e., the data that are similar to x), then c is an appropriate selection to perform the classification task for x."

For instance, the three techniques KNORA-E, KNORA-U, and OLA have customized this rule as follows.

KNORA-E: If the classifier c is able to classify at least one of the K data points that are nearest to x correctly, then it is a proper selection to carry out the classification task for x.
KNORA-U: If the classifier c is able to classify all the K data points that are the closest ones to x correctly, then it is an appropriate selection to perform the classification task for x.
OLA: The classifier which achieves the highest accuracy at the K data points that are closest to x is a proper selection to carry out the classification task for x.

- The mechanism of the proposed method is free of using the above rule since DES-ML interprets the classifier selection step as a multi-label problem in which multi-label classifiers are employed to select the appropriate classifiers.

Specifically, this article introduces a new DS technique, DES-ML, that employs the potential of a multi-label classifier so as to determine the appropriate classifiers for ensemble problems. The advantages and contributions of "DES-ML" are as follows.

- The key idea is an unprecedented DS technique formed on the multi-label classification technique suggested for ensemble problems.
- The suggested framework can perform the classifier selection task during the testing phase in a dynamic manner by employing a multi-label classifier trained in the training phase.
- The trained multi-label classifier focuses on the pool and directly picks the proper classifiers from it so that there is no need to define the local region and the competence measure as it is required to use these concepts in some other approaches such as traditional dynamic selection methods.
- Despite meta-learning techniques that use meta-features for training a traditional classifier, the process of training a multi-label classifier is performed directly via training samples.
- DES-ML is simple in implementation and its generalization phase is very fast and inexpensive.

This paper is structured as follows. Section 2 includes a stigmatization of the research background including a brief review of multi-classifier systems. Section 3 provides a detailed review of the works that form a background of the proposed method. In Sect. 4, the proposed DS technique and its mechanism are discussed. A number of experiments are performed in Sect. 5. Section 6 presents the conclusions.

2 Related works

Generally, there are two frameworks to perform classification tasks, Single Classifier System (SCS) and Multiple Classifier System (MCS). In the following, these frameworks are reviewed and their steps are described in detail.

2.1 The framework of Single Classifier Systems (SCSs) and Multiple Classifier Systems (MCSs)

Figure 2 shows the preparation process and some parts of an SCS. It can be seen from this figure that the classifier c is trained in the generation stage, then it is applied to predict the label of the input data of SCS.

Moving to Fig. 3, it is crystal clear that an MCS is composed of two parts, a pool, and a predictor. Typically, the pool contains several classifiers, and the label provided by every element in the pool is passed to the predictor as its input. Normally, the predictor is a decision fusion function such as majority voting.

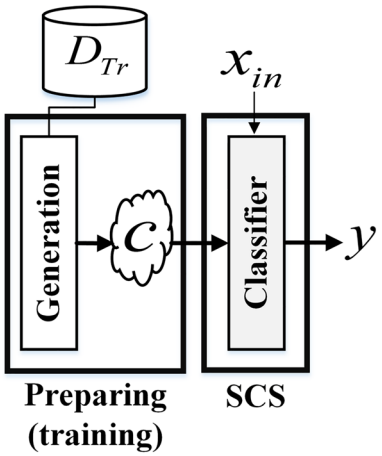

Fig. 2 Diagrammatic scheme of some parts of single classifier systems including the preparing stage

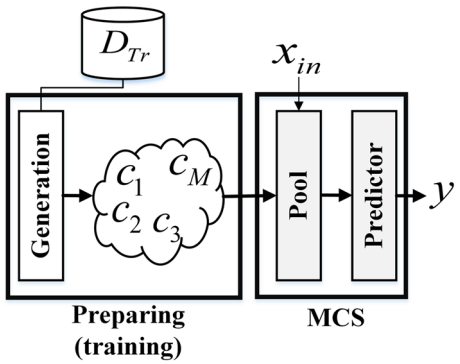

Fig. 3 Diagrammatic scheme of some parts of multiple classifier systems including the preparing stage

Similar to SCSs, the preparing process in MCSs consists of the generation stage, however, instead of producing a single classifier, MCSs generate more than one classifier as illustrated in Fig. 3. In order to create a pool including accurate and diverse classifiers to achieve high MCS performance, various techniques such as different training sets, different feature sets, and different classifier models are utilized in the generation stage [8].

- Different training sets: In this method, the training data is divided into several subsets and each classifier is trained on one of these subsets. It is a powerful and popular method and many MCSs such as bagging [33] and AdaBoost [15] have used this method.
- Different feature sets: In this method, a number of distinct representations of the data are prepared and each classifier is trained on one of these representations. In

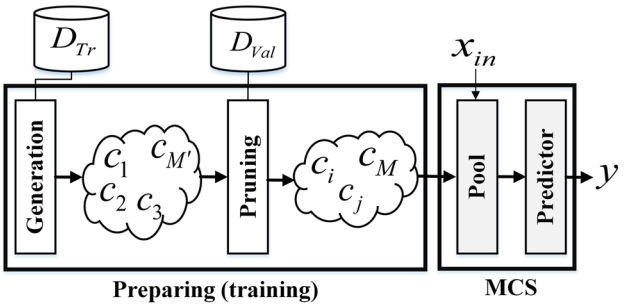

Fig. 4 Diagrammatic scheme of some parts of multiple classifier systems including the preparing stage when the static selection technique is applied

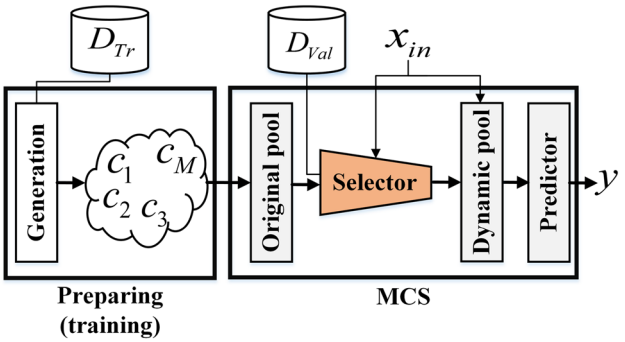

Fig. 5 Diagrammatic scheme of some parts of multiple classifier systems including the preparing stage when the dynamic selection technique is used

other words, each classifier is trained on a different feature set. Gupta et al. have recently employed this method [11].

– Different classifier models: In this method, the training data is the same for all classifiers, however, varied types of classifiers, such as decision trees and SVM, are generated. Those MCSs that use this method are called heterogeneous. Recently, Tewari and Dwivedi [38] conducted a comparative study on such systems.

Figure 4 shows a more advanced type of MCSs. In such types of systems, the outputs of the generation stage are pruned. The techniques that are used in the pruning step are known as static selection techniques. Margineantu et al. proposed this method in [28].

A more complete version of the MCS, illustrated in Fig. 5, includes another key part called a selector. A selector is responsible for the task through which the appropriate classifiers are selected. Dynamic selection (DS) techniques on which the selectors' mechanism is based normally select the appropriate classifiers in three steps: local region definition, competence level calculation, and selection. To be more specific, in the local region definition step, those pieces of data that are much the same as the original data are determined. In the next step, the classifiers' performance is computed in the local region. Finally, in the last step, the best classifiers are picked as the selector's output. Typically, two approaches are commonplace in the selection step, dynamic classifier selection (DCS) and dynamic ensemble selection (DES). In particular, DCS techniques select a single classifier only, which is the most proper classifier among the nominated ones, while DES methods determine a set of well-suited classifiers rather than choosing only a single one. It should be noted that from the perspective of the selection procedure, DS methods can be grouped into DCS and DES categories.

Since DS techniques have shown a favorable performance in classification problems, a variety of these techniques have been utilized by very many researchers in recent years. As such, K-Nearest Oracles Union (KNORA-U) [23], K-Nearest Oracles Eliminate (KNORA-E) [23], DES Performance (DES-P) [39], Overall Local Accuracy (OLA) [39] and Multiple-Classifier Behavior (MCB) [17] can be mentioned.

In the following, a detailed description of MCSs and the mechanism of selectors are provided and some applications of MCSs are described.

– Original pool: This pool consists of those classifiers that have been provided during the preparation process. For the sake of brevity, the MCS preparation process in Fig. 5 is supposed to be the same as that of typical MCSs. Nonetheless, it can have a pruning stage similar to the system illustrated in Fig. 4.
– Selector: The selector has two inputs: a set of classifiers and the input data. The selector has a duty to select the appropriate classifiers to classify the input data. The techniques used in the selector are known as dynamic selection techniques. It should be noted that a selector is the major part of an MCS.
– Dynamic pool: The selected classifiers are placed in the dynamic pool. The elements of this pool are changed for each input data. For this reason, it is called the "dynamic pool".
– Predictor: Similar to what is done in the other types of MCSs, a function such as a majority voting is employed to determine the final label of the input data.

Then, the task of the appropriate classifiers selection is carried out by making use of a DS technique. In general, given a new sample x in the testing phase, the mechanism of DS techniques to perform the classification task for x is based on Rule 1. Therefore, each DS technique consists of three basic steps. Step 1: To find the data similar to the test sample. This step is known as defining the local region in the literature. Step 2: To use criteria such as accuracy so

as to compute the competence level for each classifier with respect to the local region. More details on these two steps are provided in Sects. 2.2 and 2.3. Step 3: Lastly, among all classifiers, the ones whose performance is favorable and satisfying are selected. To be more specific, the selected classifiers are the ones whose competence level is high enough.

Finally, in the prediction stage, the selected classifiers' outputs are combined by an aggregation function such as the majority voting. In the majority voting method, the prediction of each classifier is considered as a vote, then the one whose number of votes is greater than that of all the other ones is selected as the final output prediction.

2.2 Local region definition step

The purpose of the Local Region (LR) definition step is to find those data from the D_{Val} dataset that are much the same as sample x given in the testing phase. The D_{Val}, denoted by D_{SEL} in some research papers, is a validation dataset that is usually composed of 25% of the original dataset. In summary, this step includes the process of determining a local region R_x around the sample x. It should be noted that the main techniques for dealing with the LR definition task are: KNN, clustering, and potential function. These methods are briefly described in the following.

The local region elements, determined by the KNN technique, have the smallest Euclidean distance from the sample x. However, the clustering method specifies the nearest cluster to the test sample x as the local region. The potential function is designed so that R_x includes all the items of data given in D_{Val}, however, the competence level is mostly affected by those data instances whose distance to x is less than that of the others. That is to say that a potential function is applied to model the effect of every data element on $x_j \in D_{Val}$ by the Euclidean distance between x_j and x. One of the most prevalent choices is the Gaussian potential presented in Eq. (1) [8].

$$G(x, x_j) = \exp(-d(x, x_j)^2), \text{ where, } d(x, x_j) = \left\| x - x_j \right\|_2. \quad (1)$$

We need to address the fact that the local region R_x can be defined on either the feature space (FS) or the decision space (DS). In specific, a local region in FS is composed of those data whose attributes are similar, while in DS, it includes those data points whose output profiles are much the same [13]. For a sample x, $\tilde{x} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_M\}$ indicates its output profile in which for $i = 1, 2, \ldots, M$, $\tilde{x}_i$ denotes the decision that the classifier c_i takes about x [5, 8]. Figure 6 displays both R_x and $\tilde{R}_x$ determined via FS and DS, respectively.

2.3 Competence level calculation and selection steps

It should be pointed out that given a classifier c, its competence level with respect to an instance x is equal to the performance of c in the local region R_x. In order to deal with measuring the competence level in such cases, some criteria, called competence measures, are applied. Normally, the competence measures are of two categories, the individual-based category which includes those measures for which the major information source is classifiers' individual performance, e.g., accuracy [40], and the group-based category which contains all measures whose mechanism includes the interplay of all pool members [4]. As a notable example, in order to determine the appropriate classifiers, the DS method proposed in [34] employs the diversity criterion as well as the accuracy index.

3 Background

It is noticeable that the classification task in the proposed MCS method is handled by employing multi-label classifiers and related conceptions. For this reason, before describing the proposed method, let us take a look at dynamic

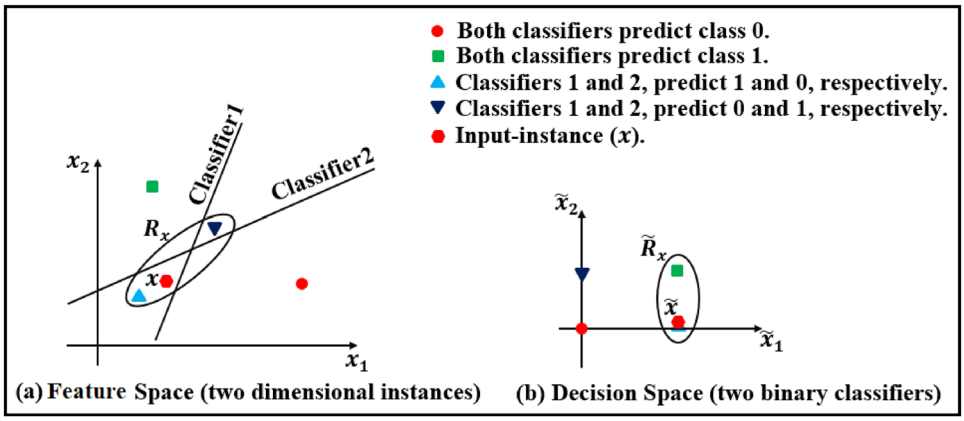

Fig. 6 The differences between feature space (FS) and decision space (DS): (a) the KNN technique (with K=2) is used in FS to select neighbors of the input instance, and (b) the KNN technique (with K=2) is utilized in DS to select neighbors of the input instance [13]

selection-based MCSs from a novel viewpoint and conduct a quick review of the essential problems considered in a multi-label mode that are closely related to the current work.

In MCSs, the main dataset is divided into three sections, D_{TR}, D_{SEL}, and D_{TE} which are applied in the generation stage to train a pool of classifiers, select the appropriate classifiers in the selection stage, and check the ability of generalization, respectively. Each dynamic selection method can also be assumed as a function, $f : x \to L_x$, in which x and L_x are respectively the input data point and the list of indices associated with those classifiers that classify the input data correctly. On the other hand, for each $x \in D_{TR}$ or $x \in D_{SEL}$, the elements of L_x are clearly specified thereby a dataset, presented in Table 2, can be formed. As Sect. 5 specifies, the dataset, presented in Table 2, is a multi-label dataset. Therefore, the function f is a multi-label classifier and any multi-label classifier such as ML-RBF can be used in order to model it.

Unlike the single-label mode, in multi-label problems, each sample is allowed to have more than a single label at the same time, which means that they are permitted to have multiple labels. To be more specific, let R^d be the input space, (i.e., $X \in R^d$), and the set $I = \{1, 2, \ldots, L\}$ is composed of the class labels. A training set in multi-label mode, D_{ML}, is defined as $D_{ML} = \{(x_i, Y_i) | x_i \in R^d, Y_i \subseteq I\}$ in which x_i and Y_i denote a sample and the collection of labels corresponding to it, respectively [41].

Example 1 Suppose that $I = \{$even, multiples of 3, multiples of 5$\}$. Then, Table 1 presents a multi-label dataset.

It is worth noting that the learning process in multi-label mode can be described as the process of inducing a function that has the ability to make predictions of a sub-collection of all possible labels associated with a new sample, and this problem is subjected to the collection of the original labels [46]. To achieve this goal, in this paper, four multi-label methods were examined including Radial Basis Function (RBF), Extreme Learning Machine (ELM) [20], Regularized Multi-label Learning method via Feature Manifold Learning (RMLFM-ELM), and K-Nearest Neighbor Multi-label Learning problem (ML-KNN) [44].

Table 1 An example of a multi-label dataset D_{ML}

i	X	Y
1	3	{multiples of 3}
2	20	{even, multiples of 5}
3	60	{even, multiples of 3, multiples of 5}
4	75	{multiples of 5}
5	99	{multiples of 3}

The ML-KNN classifier is a multi-label learning method introduced in [44] that is derived from the traditional K-nearest neighbor (KNN) algorithm. In this algorithm, for each testing sample, first, its K-nearest neighbors in the training set are identified, and then, based on the statistical information gain obtained from the label set of these neighbors, the label set for the given testing sample is determined utilizing the maximum a posteriori principle.

The RMLFM framework [30] is composed of two major stages, including a regularized multi-label classification task and local feature structure preservation. In the feature manifold technique, it is assumed that for two features that are close to each other in the original feature space, the weight vectors corresponding to these features are also close to each other. Normally, a feature graph is constructed to measure the features' local geometry. In this graph, the vertices and the weight of the edges represent the features and the affinity between the features, respectively. Next, a multi-layer ELM framework is used which consists of two learning phases including the unsupervised phase which stacks the ELM-AE technique to extract an effective feature representation, and the supervised phase which utilizes ELM as a classification task. Since the size of the data used in this paper is small, we merely utilize the supervised phase [30].

As shown in Fig. 7, the ML-RBF is a two-layer neural network whose settings are given by Statements 3 to 3 [41].

1. Input: the input is an attribute vector of dimension d.
2. Hidden layer: the hidden layer includes L collections containing prototype vectors, that is $\cup_{l=1}^{L} C_l$, where $C_l = \{c_1^l, c_2^l, \ldots, c_{k_l}^l\}$. For the lth class, c_j^l is the centroid of the jth cluster of the set $U_l = \{x_i | (x_i, Y_i) \in D, l \in Y_i\}$,

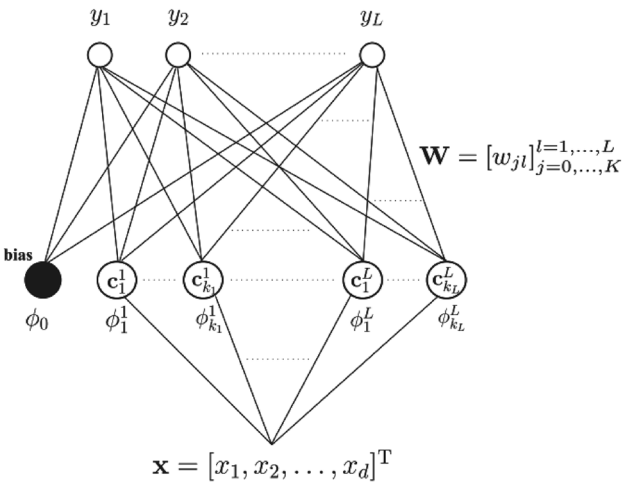

Fig. 7 The architecture of Ml-RBF neural network [41]

where $1 \leq j \leq k_l$ and $k_l = \alpha \times |U_l|$ in which α is a proportional coefficient and $|U_l|$ is the number of instances in U_l.

3. Output layer: the last layer of the Ml-RBF neural network includes L neurons and in this layer, the output of the l^{th} neuron, (i.e., yl) corresponds to the l^{th} label, (i.e., class).
4. Weights: to induce the weight matrix $W = [w_{jl}]_{(K+1)\times L}$, the mapping presented in Eq. (2) is minimized.

$$E = \frac{1}{2}\sum_{i=1}^{m}\sum_{l=1}^{L}(y_l(x_i) - t_l^i)^2, \tag{2}$$

where t_l^i and $y_l(x_i)$ are respectively the desired and the actual output of x_i associated with the lth class whose value is $+1$ in case $l \in Y_i$, otherwise, it is -1.

The ELM technique is a basic and productive single-layer feed-forward neural network (SLFN) learning algorithm that was introduced by Huang et. al [20]. Based on ELM theory, the input parameter is generated in a random manner and the output weights of the ELM method are calculated by the generalized inverse transformation. The ELM technique is a one-time learning method that provides a single optimal solution only. Major benefits of ELM are its high training rate and its strong generalization ability. It is chiefly considered a learning method proper to pattern clustering, regression, and multi-class classification. Over the last recent years, few improvements have been made to the ELM robustness. For example, Deng et al. [12] developed a regularized version of ELM (RELM) based on structural risk minimization. Notice that in the literature, the notation ELM is usually used rather than RELM. This convention is adopted throughout this paper.

In the last part of this section, some benchmark DES techniques that are compared against the proposed method in Sect. 5 are reviewed.

In OLA [39], first, the KNN technique is employed to define the local region in the feature space. Then, the accuracy of classifiers is measured in the local region. Finally, the most accurate classifier is selected. It should be noted that if there are several classifiers with the highest accuracy, one of them is selected randomly.

In the first step of the KNORA-U method [23], the local region is determined in the feature space making use of the KNN technique. Following this step, the accuracy of classifiers is measured in the local region. In the end, those classifiers are selected that correctly classify at least one data point in the local region.

The KNORA-E method [23] also applies the KNN technique first, to specify the local region in the space of attributes. Next, the accuracy criterion is employed to compute the competence level of classifiers. Eventually, those classifiers that correctly classified all the elements of the local region are selected. It is worth noting that if there is no perfect classifier (i.e. the accuracy of all classifiers is less than one), the value of K decreases. In this case, the local region needs to be redefined. In the special case where K is decreased to 0, the classifier(s) with the best accuracy is (are) selected. In DES-P [39], after defining the local region in the feature space by KNN, those classifiers whose accuracy is greater than $\frac{1}{m}$ are selected, where m is the number of classes.

The MCB [17] method first determines the region R' in the feature space by KNN technique. Next, the output profile is calculated for both the input data and $x \in R'$. Then, the similarity between the output profile of the input data and the output profile of $x \in R'$ is measured. Finally, for each $x \in R'$, it is considered as a member of the local region R_x if the similarity is greater than the threshold T. Following this step, the accuracy of classifiers is computed in R_x. In the end, the most accurate classifier is selected as the one whose accuracy is significantly better than that of the other classifiers. It should be pointed out that in case there is no classifier with accuracy far better than the accuracy of other classifiers, all elements of the pool are selected. Note that the output profile of x is a vector of labels that are predicted by all classifiers of the pool.

The META-DES [9] technique utilizes five different meta-feature groups so that every group corresponds to a varied competence measuring index that is applied in the task of classifying the input data. Figure 8 illustrates the main steps of this technique. The generation, selection, and prediction steps are the same as the standard MCS. The extraction step determines the values of meta-features with respect to the training data. Consequently, a meta-dataset is generated in this step and it is used in the meta-training step to train a meta-classifier in a way that this classifier can make predictions if the level of competency corresponding to a base classifier is sufficient to perform the task of the input sample classification in a satisfying manner. The query instance is utilized in the generalization phase in order to carry out the meta-features extraction process in the extraction step. Then, the extracted meta-features are used in the role of the meta-classifier's input to initiate the classification step. To be more specific, for each base classifier, the meta-classifier takes its level of competency into consideration to decide whether it can be included in the ensemble. In particular, META-DES considers if, for an instance, the number of votes from the winning class is close or even equal to the number of votes from the second class. To this aim, it employs a threshold h_c by which it is able to determine whether the given instance should be included in a specific class.

The DES-Hesitant technique [13] aims to create a fuzzy set $\tilde{A}$ of classifiers for which the whole pool of classifiers is considered as the universe of discourse, and the

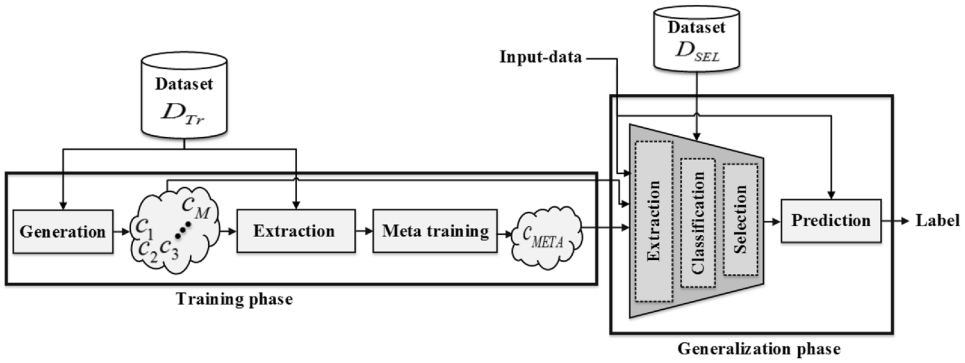

Fig. 8 The main steps of META-DES

membership degrees show the competence level of classifiers, e.g., $\tilde{A} = \left\{ \frac{0.9}{c_1}, \frac{0.8}{c_2}, \ldots \right\}$. Specifically, so as to specify the amount of the classifiers' competency level alterations, DES-Hesitant applies a number of selection rules. Next, it determines the appropriate classifiers by means of a hesitant fuzzy multiple criteria decision-making (HMCDM) technique. As seen in Fig. 9, this system uses HFDM-construction, HFS-construction, and FS-construction steps in addition to the generation, selection, and prediction steps. In the HFDM construction step, the competence level of each classifier is calculated by using a variety of criteria in different local regions, and as a result, a hesitant fuzzy decision matrix (HFDM) is constructed. Then, a fuzzy aggregation operator is applied in the HFS-construction step so that a hesitant fuzzy set (HFS) is created with respect to this decision matrix. Finally, employing a score function as well as this hesitant fuzzy set, a fuzzy set (FS) is made in the FS-construction step. In summary, DES-Hesitant performs two major tasks. First, it calculates the competence level of each classifier in different ways. Then, it creates the fuzzy set $\tilde{A}$ composed of all classifiers so that the membership degree of each classifier is its final aggregated competence level value.

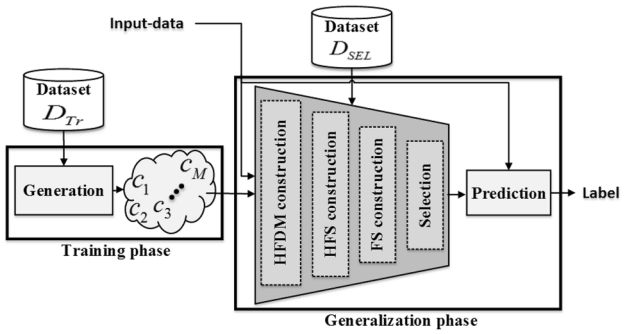

Fig. 9 The main steps of DES-Hesitant

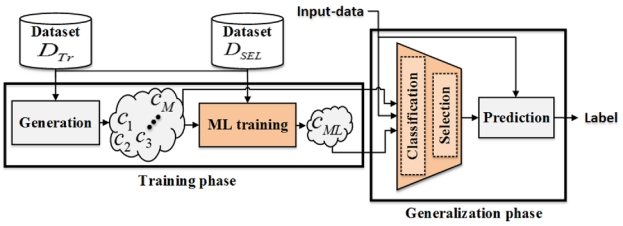

Fig. 10 The major steps of the proposed DES-ML algorithm

Table 2 The multi-label dataset D_{ML}, where $x_i \in D_{TR} \cup D_{SEL}$ and L_{x_i} is composed of indices of those classifiers which classify the instance x_i correctly

Instance (x_i)	Label (Y_i)
x_1	L_{x_1}
x_2	L_{x_2}
x_3	L_{x_3}
x_4	L_{x_4}
x_5	L_{x_5}
⋮	⋮
x_n	L_{x_n}

4 Proposed method

Figure 10 illustrates the principal phases of the proposed method. In what follows, a detailed description of DES-ML phases and steps is presented, then an example is provided to develop a deeper understanding of the suggested idea. The major steps of DES-ML are as follows.

- Generation. So as to perform the task of generating a pool, the bagging method is employed.
- ML training. The ML training step aims to train a multi-label classifier C_{ML} by using the dataset $D_{ML} = \{(x_i, Y_i) | i \in \{1, 2, \ldots, |D|\}\}$, where x_i is the input part of the dataset $D = D_{SEL} \cup D_{TR}$ and Y_i is a subset composed of the indices of those classifiers that classify x_i correctly, for example, $Y_i = \{5, 20\}$ means that the clas-

sifiers c_5 and c_{20} could classify the sample x_i in a correct manner.

- Classification. In the classification step, the vector $v = C_{ML}(x)$ is calculated, where $v(i) \in [0, 1]$ denotes the competence level corresponding to the classifier c_i.
- Selection. During this step, the Probability-Based (PB) method is employed so as to do the task of selecting the appropriate classifiers. Performing this selection process based on the PB method includes two steps. First, Eq. (3) is applied so as to assign to each classifier a selection probability according to its competence level.

$$P = \begin{cases} 0 & \text{if } C_{ML}(x) < \beta \\ C_{ML}(x) & \text{otherwise} \end{cases}, \tag{3}$$

where β is a threshold that is used to ignore the classifiers whose competence level is low. This means that a classifier whose competence level is less than β is not considered an appropriate one. Then in the second step, the Roulette wheel selection technique [26] is utilized to select the appropriate classifiers based on their selection probability.

- Prediction. The majority voting method is applied to aggregate the votes associated with the selected classifiers. The parameters of the Roulette wheel technique are set as follows.
  - The probability vector, P, is taken as the output of the multi-label classifier C_{ML}.
  - The number of iterations $= W \times M$, where $W = \frac{7n}{2M}$, M indicates the pool length and n denotes the number of non-zero elements in P.

For the sake of describing the DES-ML method in an accurate and detailed manner, a notable example is given in the following.

Example 2 Generation. Suppose that a set of five classifiers $\{c_1, c_2, c_3, c_4, c_5\}$, displayed in Fig. 11, are generated based on the bagging technique.

ML training. Consider dataset D which is composed of 14 elements, represented in Fig. 11 by blue triangular and red circular shapes. The blue triangular samples display class 0 (or -1) and the red circular ones characterize class 1. The multi-label dataset D_{ML}, presented in Table 3, is constructed according to the dataset D and the classifiers c_1, c_2, c_3, c_4 and c_5 which are illustrated in Fig. 11. In this figure, there are two types of arrows, blue ones, and red ones, for the lines (classifiers). The colors indicate the class that is predicted by the corresponding side of the classifier. For instance, considering x_1, the classifiers that predict its label correctly are c_1, c_3 and c_4, thereby $Y_1 = [+1, -1, +1, +1, -1]$. In this step, it

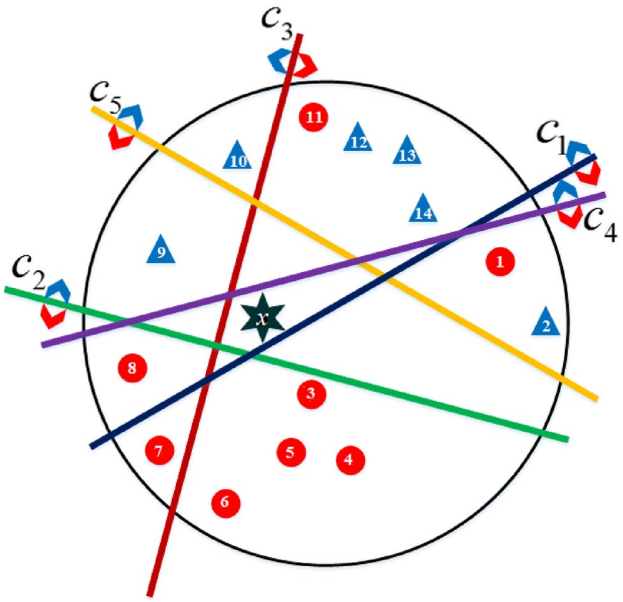

Fig. 11 The dataset D and the classifiers c_1, c_2, c_3, c_4 and c_5 are generated by the bagging technique in the generation step. The samples are represented by numbers 1 to 14

Table 3 The multi-label dataset D_{ML} is constructed based on Fig. 11 and is used for training the classifier c_{ML}

x_i	Y_i
x_1	$[+1, -1, +1, +1, -1]$
x_2	$[-1, +1, -1, -1, +1]$
x_3	$[+1, +1, +1, +1, +1]$
x_4	$[+1, +1, +1, +1, +1]$
x_5	$[+1, +1, +1, +1, +1]$
x_6	$[+1, +1, +1, +1, +1]$
x_7	$[+1, +1, -1, +1, +1]$
x_8	$[-1, +1, -1, +1, +1]$
x_9	$[+1, +1, +1, +1, -1]$
x_{10}	$[+1, +1, +1, +1, +1]$
x_{11}	$[-1, -1, +1, -1, -1]$
x_{12}	$[+1, +1, -1, +1, +1]$
x_{13}	$[+1, +1, -1, +1, +1]$
x_{14}	$[+1, +1, -1, +1, +1]$

is supposed that the multi-classifier C_{ML} is trained over the examples given in Table 3.

Selection. Assume that $\beta = 0.3$ and $C_{ML}(x) = [0.6, 0.2, 0, 0.8, 0.5]$. As a result, the probability vector is calculated as $P = [0.6, 0, 0, 0.8, 0.5]$. It should be noted that since the competence level of the classifier c_2 is less than the threshold β, $P(2)$ is set to zero. The probability vector P is passed down to the Roulette wheel method which is employed to handle the selection task according to the mechanism described in the following.

– The selection probability P' is calculated based on P by the formula given in Eq. (4).

$$P'(i) = \frac{\sum_{j=1}^{i} P(j)}{S}, \tag{4}$$

where $S = \sum_{i=1}^{\text{Length}(P)} P(i)$. Using Eq. (4), the selection probability, P', for this example is calculated as $P' = \left[\frac{0.6}{1.9}, \frac{0.6}{1.9}, \frac{0.6}{1.9}, \frac{1.4}{1.9}, \frac{1.9}{1.9}\right] = [0.31, 0.31, 0.31, 0.73, 1]$.
– K random numbers are generated in the interval $[0, 1]$. In this example, it is assumed that $K = 8$ and the associated eight random numbers are $rnd = [0.4, 0.5, 0.6, 0.7, 0.8, 0.5, 0.9, 0.4]$.
– It is supposed that I is the empty list. For each $r \in rnd$, the following conditions are checked:

  – if $r < P'(1)$, append the index 1 to I;
    else if $r < P'(2)$, append the index 2 to I;
    else if $r < P'(3)$, append the index 3 to I;
    else if $r < P'(4)$, append the index 4 to I;
    else if $r < P'(5)$, append the index 5 to I.

  As a result, I is calculated as $I = [4, 4, 4, 4, 5, 4, 5, 4]$.
– Finally, in case $i \in I$, the classifier c_i is selected. Therefore, the classifiers c_4 and c_5 are selected in order to perform the task of predicting the testing example label during the prediction step.

Prediction: Consider the test sample x indicated by a black star in Fig. 11. At a glance, it is crystal clear that $c_4(x) = c_5(x) = +1$ since the star sample is located in the red side of the classifiers $c_4(x)$ and $c_5(x)$. Hence, for the testing instance x, the label $+1$ is predicted.

In summary, the proposed method deals with the DES problem by turning it into a multi-label classification problem, where the given data plays a substantial role in the proposed solution. It should be noted that one of the main challenges which many DES methods encounter is to define a criterion for measuring the competence level of classifiers. In this respect, first, a local region is normally defined around the input data. Then, the performance of classifiers in this region is considered as their competence level. However, the proposed method is not dependent on such a constraint and determines the competence level of classifiers by formulating it as a new classification problem. This new problem use neither the local region nor the criteria to measure the level of classifiers' competence. In other words, the proposed algorithm passes the input data directly to the multi label and uses the corresponding output to select the classifiers. Therefore, the progress in the field of multi-label-classifiers help tackles the task of finding the optimal solution for DES problems.

5 Experimental results and discussion

The current section presents the numerical results of experiments performed over 34 datasets for justifying the performance of the suggested technique. In order to carry out the simulations and to perform experiments on the datasets, the python programming language and DESlib [7] library are employed. This library gives implementations of several benchmark DS methods so as to draw comparisons between them and the DES-ML technique. These methods are introduced in Sect. 5.2.

5.1 Datasets

The proposed method is examined on thirty-four benchmark datasets whose characteristics including their name, dimension, source repository, and numbers of classes and instances can be found in Table 4.

As mentioned earlier, in DS techniques, each dataset is divided into three parts, training, selection, and testing which are denoted by D_{Tr}, D_{Val} and D_{Te}, respectively. In this paper, the policy for dividing datasets is as follows:

– D_{Tr}: 50%;
– D_{Val}: 25%;
– D_{Te}: 25%.

5.2 Comparison methods

The benchmark techniques applied to numerical experiments include the nine following methods.

1. Overall Local Accuracy (OLA) [39]: This method selects the most accurate classifiers. In case the difference between the competence level of the base classifiers is less than the threshold T_{dif}, their performances are considered equivalent. In this technique, the KNN and accuracy criteria are used respectively to define the local region in the feature space and to measure the competence level of classifiers.
2. K-Nearest Oracles Union (KNORA-U) [23]: In this method, those classifiers are selected whose accuracy is equal to or more than the selection threshold T_{sel}. It employs KNN to determine the local region in the feature space.
3. K-Nearest Oracles Eliminate (KNORA-E) [23]: In this technique, KNN is utilized to specify the competence level of classifiers. It only selects the perfect classifiers that are the ones for which $T_{sel} = 1$.

Table 4 Characteristics of the datasets used in the experiments

Dataset ID	Dataset Names	Instances	Dimension	Classes	Source
D1	Seeds	210	8	3	UCI https://archive.ics.uci.edu/ml/index.php
D2	Waveform	5000	41	3	UCI
D3	Wall-Robot	5456	25	4	UCI
D4	Wine	178	13	3	UCI
D5	Synthetic-Control	600	62	6	UCI
D6	CMC	1473	10	3	UCI
D7	Segment	2310	20	7	UCI
D8	Balance-Scale	625	5	3	UCI
D9	Vehicle	846	19	4	UCI
D10	Cnae	1080	857	9	UCI
D11	Glass	214	9	6	UCI
D12	Cardiotocography	2126	36	10	UCI
D13	Semeion	1593	257	10	UCI
D14	Mfeat-Pixel	2000	241	10	UCI
D15	Mfeat-Karhunen	2000	65	10	UCI
D16	Mfeat-Fourier	2000	77	10	UCI
D17	Mfeat-Zernike	2000	48	10	UCI
D18	Mfeat-Factors	2000	217	10	UCI
D19	Mfeat-Morphological	2000	7	10	UCI
D20	Optdigits	5620	65	10	UCI
D21	Breast-Cancer	863	10	2	UCI
D22	Liver Disorders	345	6	2	UCI
D23	Vertebral Column	310	6	2	UCI
D24	Haberman's Survival	306	3	2	UCI
D25	Heart	270	13	2	UCI
D26	Ionosphere	351	34	2	UCI
D27	Pima	768	8	2	UCI
D28	Sonar	208	60	2	UCI
D29	Transfusion	748	4	2	UCI
D30	Boston-Corrected	506	21	2	OPENML https://www.openml.org/
D31	Car-Evaluation	1728	22	4	OPENML
D32	Teaching-Assistant	151	7	3	OPENML
D33	Australian	690	14	2	OPENML
D34	KC2	522	22	2	OPENML

4. DES Performance (DES-P) [39]: In DES-P, the local region is created in the feature space by means of KNN. Then, classifiers whose accuracy is greater than the selection threshold T_{sel} are selected.
5. Multiple-Classifier Behavior (MCB) [17]: In MCB, making use of KNN, first a region is defined in the feature space. Next, this region is pruned in the decision space. Finally, the most accurate classifier is selected. The performances of the two base classifiers are considered equivalent, in case the difference between their competence level is less than the difference threshold T_{dif}.
6. META-DES [9]: This method applies the competence level criteria in two local regions. These regions are specified by KNN with K and K_p elements in the feature space and the decision space, respectively. The META-DES uses two main thresholds, the sample selection threshold h_c which is utilized to select samples for meta-feature extraction, and the classifier selection threshold T_{sel} which is employed to determine the appropriate classifiers.
7. DES-Hesitant [13]: In this technique, the competence of classifiers is calculated in two local regions with K and K_p elements in the feature space and the decision

space, respectively. Then, a fuzzy set composed of the appropriate classifiers is created. Finally, a classifier c is selected if its membership degree is greater than the selection threshold T_{sel}.

8. "Bagging+": This method is the so-called bagging algorithm which is trained using $D_{train} + D_{val}$.
9. "Bagging": This method is also the bagging algorithm, but it is trained only on D_{train}.

5.3 Parameters settings

The process of setting parameters in this paper was performed in a straightforward manner since the proposed method required no parameter that is prevalent in dynamic selection techniques, such as the number of local regions, the number of elements in each region, the local region definition space and the competence level criterion. In the following, the best values of parameters and configurations for various parts of the comparison methods are given as well as those of the proposed method. Some datasets have been used to tune the parameters and to determine their values.

Generation and prediction stages setups: One of the key issues in the Generation phase is the diversity of the pool. The major reason is that the diversity of the pool is a chief cause of increasing the generalization of DES algorithms. The bagging technique that is usually used in the Generation phase is based on the following policy: "Train each classifier using a separate dataset to guarantee the diversity of the pool". Specifically, the bagging technique randomly divides the training data into M folds with replacement, where M is the number of the classifiers included in the pool. Then, each of the classifiers is trained by making use of one of these folds. In this paper, the bagging algorithm is used to generate a pool of classifiers (weak classifiers). This pool consists of 100 single-layer perceptrons. A single-layer Perceptron is a linear neural network for which the number of the input neurons equals the number of the features, and the number of the output neurons equals one and n for the two-class and multi-class datasets, respectively, where n is the number of classes. This NN predicts the label of the instance x according to the number of the classes. Specifically, the data class in a two-class dataset will be one if the output of the NN is greater than zero, otherwise, it will be zero.

$$y_x = \begin{cases} 1 \text{ if } \mathrm{NN}(x) \geq 0 \\ 0 \text{ otherwise} \end{cases},$$

where x is the input data, y_x is the predicted label and $\mathrm{NN}(x)$ is the output of the NN. In multi-class datasets, the label of the input data, y_x, is calculated by Eq. (5).

$$y_x = \mathrm{argmax}(\mathrm{NN}(x)). \tag{5}$$

ML training and DS selection setups: In the ML training step, the dataset D_{ML} is utilized to produce C_{ML} by training an ML-RBF network. So as to select the appropriate classifiers in the selection step, the Probability-Based (PB) method is employed.

General Settings: For all the DS techniques except for the proposed method, the number of the nearest neighbors in the feature space, K, is fixed as seven on all the datasets. In MCB, META-DES, and DES-Hesitant techniques, the number of the nearest neighbors in the decision space, K_p, is set to five on all the datasets. The difference threshold, T_{diff}, is fixed as 0.1 in OLA and MCB. The classifier selection threshold, T_{sel}, is set to $\frac{1}{K}$, 1, $\frac{1}{m}$, 0.5, and 0.8 for KNORA-U, KNORA-E, DES-P, META-DES, and DES-Hesitant, respectively, in which K is the local region size and m is the number of the classes. The sample selection threshold, h_c, is set to 0.7 in the META-DES method.

The proposed method uses hyper-parameters β and W. The efforts to provide a proper setting for these parameters were initiated by assigning a fixed number to them as follows: $\beta = 0.4, 0.8$ and $W = 1, 0.6$. Checking the output of DES-ML on some datasets, it appeared that the fixed parameters could not be appropriate for all the datasets. Therefore, the authors made attempts to heuristically investigate how these parameters could depend on the numbers of the instances and classes corresponding to each dataset or on the output of C_{ML} (e.g. mean and standard deviation). For example, a typical formula that was examined to calculate β is given in Eq. (6).

$$\beta = 0.7 \times (n-1) \quad \text{and} \quad \beta = min(P) + 0.5 \times (max(P) - min(P)), \tag{6}$$

where n is the number of the classes.

Eventually, the hyper-parameters were suggested as $\beta = \mu_P + \sigma_P$ in which μ_P and σ_P denote the mean and standard deviation associated with the probability vector, P, respectively. The number of the generated random numbers, W, was determined as $W = \frac{7n}{2M}$ for which M indicates the pool length and n denotes the number of non-zero elements in P. It is noticeable that the proposed method does not use the local region and competence measure conceptions. For this reason, parameters such as K or K_p, relevant to these notions, are not included in this method.

In the following, the framework of the proposed method, DES-ML, is described. Generally, the output of the multi-label classifier C_{ML} is a vector $P = [p_1, p_2, \ldots, p_M]$, where M is the size of the pool and p_i corresponds to the competence level of the classifier c_i and indicates the chance to select c_i. In other words, those classifiers for which p_i is higher are selected. This issue makes the DES-ML method capable of predicting the label of most input instances correctly. In specific, more than 50% of the classifiers selected by DES-ML are highly appropriate. According to the experimental

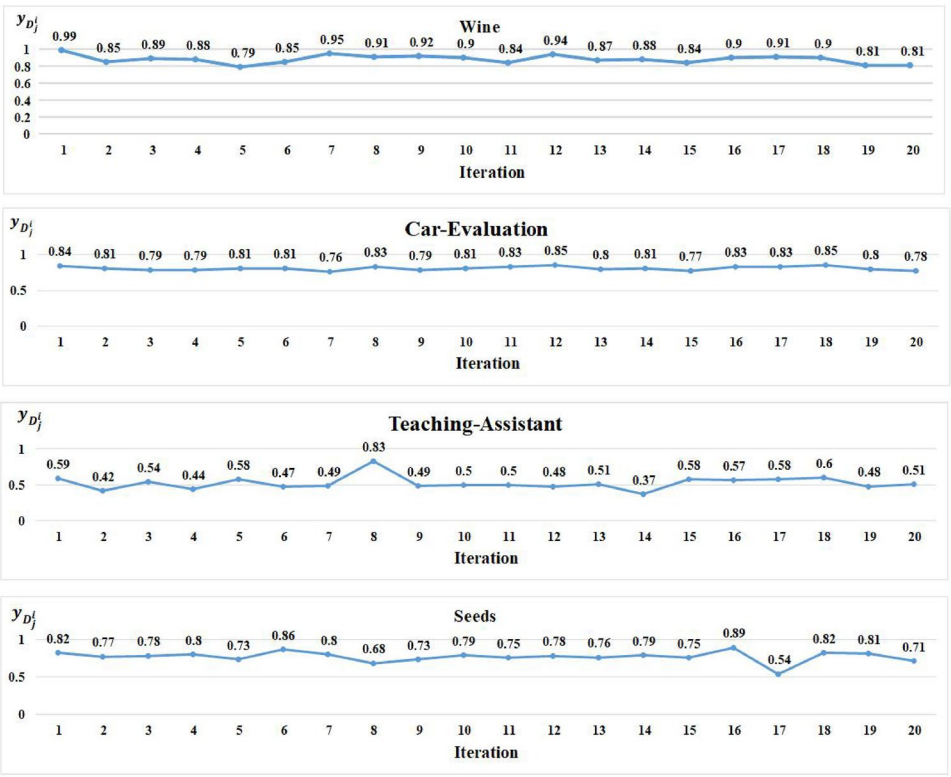

Fig. 12 The mean ratio of the number of classifiers that are selected correctly by C_{ML} to the total number of the selected classifiers

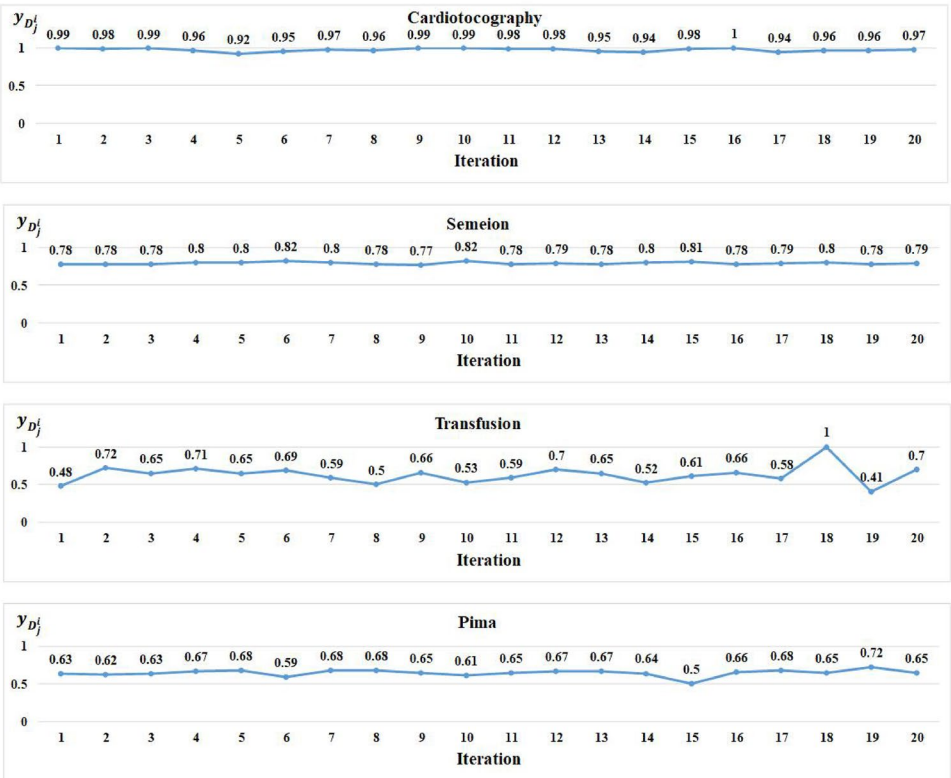

Fig. 13 The mean ratio of the number of classifiers that are selected correctly by C_{ML} to the total number of the selected classifiers

results given in Tables 5 and 6, the DES-ML method showed a strong performance compared to the benchmark methods. This means that the C_{ML} classifier is highly effective.

Figures 12 and 13 illustrate the mean ratio of the classifiers that are correctly selected by C_{ML}. In this figure, the x and y axis indicate the iteration and the mean ratio, respectively. The mean ratio is calculated by Eq. (7).

$$y_{D_j^i} = \frac{1}{|D_j^i|} \sum_{x \in D} \frac{S_{true}^x}{S_{all}^x}, \tag{7}$$

where S_{true}^x is the number of those classifiers that are selected correctly, S_{all}^x is the total number of the classifiers selected to classify the instance x, D_j^i is the testing data corresponding to the jth dataset in the ith iteration and $|D_j^i|$ is the size of D_j^i. The mean of $y_{D_j^i}$ is calculated using Eq. (8), where rep indicates the number of iterations for each algorithm. Similar to [9], in this paper, rep is set to 20. Table 5 presents the accuracy of all the algorithms. This quantity is calculated by averaging over all 20 iterations.

In Fig. 14, the vertical axis, y_{D_j}, represents the mean $y_{D_j^i}$ given by Eq. (8).

$$y_{D_j} = \frac{1}{rep} \sum_{i \in \{1,2,\ldots,rep\}} y_{D_j^i}. \tag{8}$$

Note: All the parameters of the methods discussed in this paper depend on the vector P which is the output of the multi-label classifier C_{ML} that is produced based on each testing data. Hence, σ_P, μ_P, and n which denote the standard deviation, the mean, and the number of non-zero elements of P, respectively, are changing concerning the testing data. Since β depends on both σ_P and μ_P, and W relies on n thereby these parameters take different values for each input data.

5.4 Results and comparisons

For the sake of justifying the performance of DES-ML in comparison to that of the benchmark methods, many experiments were performed. In this section, the results of these experiments are discussed.

To perform the experiments, first, each dataset is divided into three disjoint parts in a random manner including D_{te}, D_{tr}, and D_{sel}. Next, the bagging method (the bagging mechanism is similar to the k-fold cross-validation except for the fact that the folds are randomly selected with replacement) is utilized to select a number of the data from D_{tr} to train the base classifiers. Therefore, the two above steps are repeated 20 times where the same partitions are used for all methods. The average accuracy is reported in Table 5. According to Table 5, the proposed method achieves the highest accuracy over datasets D1, D2, D6, D9, D12, D14, D15, D21, D22, D23, D24, D29, D31, D32, D33, and D34. Normally, the accuracy of this method is close to that of the state-of-the-art techniques on the other datasets.

To have a scientific conclusion and comparison, the statistical non-parametric Quade test is conducted over the results of Table 5. The average rank values of all the methods given by the Quade test are presented in Table 6. According to the results of this table, our proposed method (DES-ML) has the best rank. Notice that in Quade test ranking, lower values present better ranks. To demonstrate the superiority of the DES-ML technique, a rampant type of post hoc procedure is conducted for comparing the methods according to their rank values. Consequently, Li procedure is utilized for this purpose and the corresponding results are given in Table 7. The p values of the Li procedure are given in the last column of Table 7. The control method is DES-ML so that the rest of the techniques are compared with it in a pairwise manner. In each of the pair-wise comparisons, the default hypothesis (null hypothesis) is that the compared methods are considered equal in a statistical manner, while the alternative hypothesis is that the compared approaches are different significantly. Therefore, the two compared methods will have a significant difference if the default hypothesis is rejected. According to the last column of Table 7, the default hypotheses are rejected when $p_{Li} \leq 0.028758$. It is concluded that DES-ML outperforms the other methods except for DES-Hesitant.

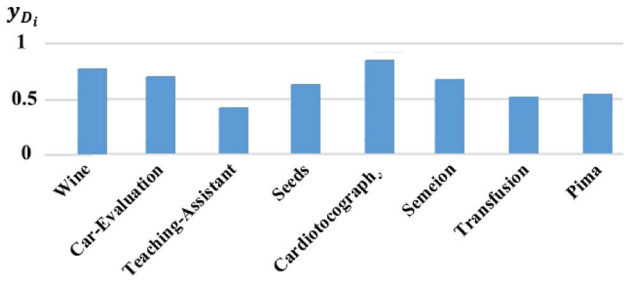

Fig. 14 Mean of $y_{D_j^i}$ over the datasets used

6 Concluding remarks

Far as we know, the DES-ML technique can be considered the first and the only approach that uses multi-label classification concepts so as to select appropriate classifiers for the testing data. In spite of traditional dynamic selection methods that apply local regions and competence measures for selecting appropriate classifiers, the proposed approach selects the classifiers by employing a multi-label classifier, thereby a novel framework is established for the evaluation of dynamic selection techniques. Regularly, DS techniques

Table 5 The average accuracy obtained by the proposed DES-ML method and the existing DS systems in the literature over D_{Te} for 20 times of run. A pool of 100 perceptrons considered as the base classifiers is used for all the techniques. The best results are highlighted in boldface

Datasets ID	DES-ML-ELM	DES-ML-RBF	DES-ML-RMLFM	DES-ML-KNN	DES-Hesitant	KNORA-U	KNORA-E	DES-P	OLA	MCB	META-DES	Bagging	Bagging+
D1	**0.9476**	0.9454	0.9432	0.9385	0.9413	0.9414	0.9453	0.9398	0.9121	0.9105	0.9421	0.9405	0.9443
D2	**0.8576**	0.849	0.849	0.8485	0.8488	0.8472	0.8373	0.8461	0.8267	0.818	0.8464	0.8496	0.8478
D3	0.8048	0.7398	0.7972	0.7826	0.808	0.7477	**0.8242**	0.7296	0.8054	0.8035	0.8076	0.6485	0.6695
D4	0.9705	0.9716	0.9773	**0.9841**	0.9773	0.9773	0.9761	0.9761	0.9557	0.9523	0.9739	0.9739	0.9773
D5	0.97	0.9657	**0.9753**	0.9607	0.9707	0.9603	0.972	0.9583	0.9453	0.9453	0.96	0.9593	0.9693
D6	**0.512**	0.4948	0.5095	0.5106	0.4815	0.4807	0.4626	0.4864	0.4806	0.4734	0.4765	0.4526	0.4719
D7	0.9504	0.9395	0.952	0.9492	0.9502	0.9386	**0.9557**	0.9348	0.9445	0.9434	0.9492	0.9289	0.9309
D8	0.9071	0.9131	**0.9186**	0.9096	0.908	0.8901	0.9016	0.8878	0.9019	0.8968	0.9119	0.8865	0.8891
D9	**0.8128**	0.778	0.7915	0.7872	0.7668	0.7555	0.7704	0.7464	0.7562	0.7545	0.7687	0.7438	0.7652
D10	0.9048	0.8998	0.9015	0.8941	0.9094	0.9024	0.9028	0.9006	0.8619	0.85	0.9109	0.9019	**0.9252**
D11	0.6226	0.6358	0.6264	0.6264	**0.6434**	0.6057	0.6292	0.5943	0.6142	0.617	0.6208	0.5651	0.6047
D12	**0.9998**	0.9992	0.9994	0.9994	0.9993	0.9994	0.9994	0.9994	0.9979	0.997	0.9996	0.9996	0.9995
D13	0.9178	0.9173	0.9196	0.9121	0.9123	0.9132	0.9182	0.9103	0.8857	0.8764	0.9152	0.9129	**0.9211**
D14	**0.9684**	0.9643	0.9654	0.9594	0.964	0.9628	0.9656	0.9621	0.9463	0.9433	0.9633	0.9626	0.9645
D15	**0.9476**	0.9454	0.9432	0.9414	0.9413	0.9414	0.9453	0.9398	0.9121	0.9105	0.9421	0.9405	0.9443
D16	0.798	0.8051	0.7902	0.8	0.801	0.8004	0.7882	0.7952	0.7634	0.7561	0.8027	0.7981	**0.8125**
D17	0.805	0.8209	0.8108	0.8122	0.8183	**0.8248**	0.8137	0.8233	0.7966	0.7908	0.8203	0.8213	0.8193
D18	0.971	0.968	0.9712	0.9658	0.9691	0.9677	0.9696	0.9674	0.9555	0.9547	0.9673	0.9667	**0.9721**
D19	0.742	**0.7443**	0.7242	0.7406	0.7379	0.7316	0.706	0.7189	0.728	0.7206	0.7334	0.7369	0.7117
D20	0.9695	0.9654	0.9685	0.9675	0.9671	0.9654	**0.971**	0.9643	0.9567	0.9536	0.9648	0.9636	0.9678
D21	**0.9725**	0.9585	0.9684	0.9637	0.9661	0.9602	0.9608	0.9602	0.9573	0.9579	0.9588	0.9553	0.9637
D22	**0.6895**	0.6686	0.6593	0.6407	0.6512	0.639	0.6506	0.6558	0.6634	0.6599	0.6326	0.5779	0.657
D23	**0.9476**	0.9454	0.9432	0.7462	0.9413	0.9414	0.9453	0.9398	0.9121	0.9105	0.9421	0.9405	0.9443
D24	**0.7724**	0.7151	0.7171	0.7276	0.6934	0.7191	0.6559	0.7263	0.7125	0.7079	0.7296	0.7316	0.6901
D25	0.8269	0.8157	0.8254	**0.8463**	0.7963	0.8097	0.8127	0.8082	0.7948	0.7903	0.8134	0.8052	0.8216
D26	0.8852	0.8807	0.8682	0.8739	0.8688	0.8778	0.8852	0.8744	0.8665	0.8551	0.8682	0.8415	**0.8886**
D27	0.7589	0.7539	**0.7635**	0.7599	0.7195	0.7469	0.7143	0.7534	0.7164	0.7081	0.7253	0.7234	0.7169
D28	0.7154	0.6481	0.6	0.6346	0.7702	0.7712	**0.775**	0.7644	0.7529	0.7625	0.6288	0.5269	0.775
D29	**0.7733**	0.7639	0.7594	0.7594	0.7203	0.75	0.7104	0.7602	0.7561	0.7599	0.7553	0.7586	0.7134
D30	0.8667	0.8774	**0.8889**	0.8619	0.8655	0.8615	0.869	0.8679	0.8544	0.8544	0.8587	0.8488	0.8552
D31	**0.9417**	0.9243	0.9375	0.9317	0.925	0.9098	0.9321	0.9042	0.9141	0.9087	0.9219	0.9013	0.9091
D32	**0.5184**	0.4697	0.4868	0.4789	0.4684	0.4789	0.4947	0.4868	0.4842	0.4724	0.4197	0.4053	0.4697
D33	**0.8628**	0.8422	0.8488	0.8401	0.8366	0.8468	0.841	0.8456	0.8334	0.8262	0.8352	0.8096	0.8331
D34	**0.8338**	0.8212	0.8054	0.8246	0.8027	0.825	0.7969	0.8242	0.8008	0.7942	0.8158	0.8004	0.8123

The best average accuracy obtained by the proposed DES-ML method and the existing DS systems in the literature over D_{Te} for 20 times of run are highlighted (by bold face)

Table 6 Average rank values of the proposed method and the state-of-the-art methods obtained based on the Quade test

Algorithm	Ranking
DES-ML-ELM	3.0840
DES-ML-RBF	5.0718
DES-ML-RMLFM	5.1882
KNORA-E	6.2000
DES-ML-KNN	6.5248
DES-Hesitant	6.6824
KNORA-U	7.0294
META-DES	7.0420
Bagging+	7.0731
DES-P	7.7639
OLA	9.1622
Bagging	10.0311
MCB	10.1471

Table 7 p values obtained by applying the Li post hoc method to the results of the Quade test for $\alpha = 0.05$. Default hypotheses with $p_{Li} \leq 0.028758$ are rejected. The proposed method is used as the control method in this table

i	Algorithm	$z = (R_0 - R_i)/\text{SE}$	p value	p_{Li}
12	MCB	2.662868	0.007748	0.028758
11	Bagging	2.619147	0.008815	0.028758
10	OLA	2.291555	0.021931	0.028758
9	DES-P	1.764368	0.077670	0.028758
8	Bagging+	1.503942	0.132596	0.028758
7	META-DES	1.492220	0.135642	0.028758
6	KNORA-U	1.487468	0.136891	0.028758
5	DES-Hesitant	1.356621	0.174902	0.028758
4	DES-ML-KNN	1.297217	0.194556	0.028758
3	KNORA-E	1.174767	0.240088	0.028758
2	DES-ML-RMLFM	0.793316	0.427594	0.028758
1	DES-ML-RBF	0.749437	0.453594	0.050000

work based on the intuition that "a classifier c which can efficiently classify the data located in a local region (i.e., the data that is similar to a datapoint x) is an appropriate classifier to do the classification task for x". This rule imposes some constraints on the mechanism of dynamic selection techniques such as the necessity of defining local regions and employing competence measures. Therefore, two major limitations of dynamic selection techniques are to adopt a proper approach to determine a local region and to select well-suited criteria to measure the competence level of classifiers in the local region. To tackle these issues in a satisfying manner, this paper proposed an innovative dynamic selection technique resting on the idea that multi-label conceptions can develop the capacity to relax the above rule and its constraints so that dynamic selection problems are solved without the need to define a local region and to apply criteria for measuring the competence level of classifiers. In this paper, 34 datasets have been used to compare the proposed method with other popular dynamic selection techniques. At the start point of this research, the authors' conjecture was formulated as the hypothesis that the problem of selecting different classifiers for classifying a pattern through dynamic selection approaches is more likely to be put into a multi-label classification framework in an inherent manner. The conducted experiments and their outcomes strongly confirmed this hypothesis. The numerical results corresponding to the experimental study clearly demonstrate the efficiency of the proposed approach. Considering the widespread usage of MCS, the proposed method can be utilized in such systems to enhance their efficiency in applications. It is remarkable that the proposed method can be simply implemented since its only requirements are to train a multi-label classifier C_{ML} in the training phase and to apply C_{ML} to classify the input data in the generation phase.

A potential direction for future works is to research methods that have the ability to produce efficient multi-label datasets so as to train the multi-label classifiers applied in the training phase. It is also worth studying multi-label classifiers to find those ones that can be used efficiently for the purpose of handling the training phase. Furthermore, when it comes to training C_{ML}, it is particularly significant to find a feature set that produces incredibly strong performance. Thus, making use of feature selection techniques to determine the best feature set according to the set of labels is an interesting problem to be dealt with. Another forward-looking idea is to utilize the capacity of deep neural networks to solve dynamic selection problems without using local regions and competence level criteria. Moreover, it is noteworthy that the application of the proposed approach to the ensemble learning problem in deep neural networks is a prospective question that can be pondered. It is also interesting to study the proposed method in case the same percentage is given for the full ensemble (Figs. 12, 13, and 14) and to investigate if this issue can improve the proposed method in terms of better integration of the selected classifiers in the prediction step of MCS.

References

1. Balogun A, Bajeh A, Mojeed H, Akintola A (2020) Software defect prediction: a multi-criteria decision-making approach. Nigerian J Tech Res 15(1):35–42
2. Bedi P, Gupta N, Jindal V (2020) I-siamids: an improved siam-ids for handling class imbalance in network-based intrusion detection systems. Appl Intell 51:1133–1151
3. Bhatore S, Mohan L, Reddy YR (2020) Machine learning techniques for credit risk evaluation: a systematic literature review. J Banking Fin Tech 4:111–138
4. Britto S Jr, A., Sabourin R, Soares de Oliveira L, (2014) Dynamic selection of classifiers-a comprehensive review. Pattern Recogn 47:3665–3680. https://doi.org/10.1016/j.patcog.2014.05.003

5. Cavalin PR, Sabourin R, Suen CY (2013) Dynamic selection approaches for multiple classifier systems. Neural Comput Appl 22(3):673–688
6. Choudhary R, Shukla S (2021) A clustering based ensemble of weighted kernelized extreme learning machine for class imbalance learning. Expert Syst Appl 164:114041. https://doi.org/10.1016/j.eswa.2020.114041
7. Cruz RMO, Hafemann LG, Sabourin R, Cavalcanti GDC (2020) Deslib: a dynamic ensemble selection library in python. J Mach Learn Res 21(8):1–5
8. Cruz RM, Sabourin R, Cavalcanti GD (2018) Dynamic classifier selection. Inf Fusion 41(C):195–216. https://doi.org/10.1016/j.inffus.2017.09.010
9. Cruz RM, Sabourin R, Cavalcanti GD, Ren TI (2015) META-DES: a dynamic ensemble selection framework using meta-learning. Pattern Recogn 48(5):1925–1935
10. Das A, Ghosh S, Thunder S, Agarwal S, Chakrabarti A (2020) Automatic covid-19 detection from x-ray images using ensemble learning with convolutional neural network. Europe PMC. https://doi.org/10.21203/rs.3.rs-51360/v1
11. Das Gupta J, Samanta S, Chanda B (2018) Ensemble classifier-based off-line handwritten word recognition system in holistic approach. IET Image Process 12(8):1467–1474
12. Deng W-Y, Zheng Q-H, Chen L, Xu X-B (2010) Research on Extreme Learning of neural networks. Chinese J Comput 33(2):279–287
13. Elmi J, Eftekhari M (2020) Dynamic ensemble selection based on hesitant fuzzy multiple criteria decision making. Soft Comput 24:12241–12253
14. Fan R, Feng R, Wang L, Yan J, Zhang X (2020) Semi-mcnn: a semisupervised multi-cnn ensemble learning method for urban land cover classification using submeter hrrs images. IEEE J Selected Topics Appl Earth Observ Remote Sens 13:4973–4987
15. Freund Y, Schapire RE (1997) A decision-theoretic generalization of on-line learning and an application to boosting. J Comput Syst Sci 55(1):119–139. https://doi.org/10.1006/jcss.1997.1504
16. García S, Zhang ZL, Altalhi A, Alshomrani S, Herrera F (2018) Dynamic ensemble selection for multi-class imbalanced datasets. Info Sci 445:22–37
17. Giacinto G, Roli F (2001) Dynamic classifier selection based on multiple classifier behaviour. Pattern Recogn 34:1879–1881
18. Gomes HM, Barddal JP, Enembreck F, Bifet A (2017) A survey on ensemble learning for data stream classification. ACM Comput Surv (CSUR) 50(2):1–36. https://doi.org/10.1145/3054925
19. Hou WH, Wang XK, Zhang HY, Wang JQ, Li L (2020) A novel dynamic ensemble selection classifier for an imbalanced data-set: An application for credit risk assessment. Knowl-Based Syst 208:106462
20. Huang G-B, Zhu, Q-Y, Siew, C-K (2006) Extreme learning machine: theory and applications, pp 489–501
21. Jin B, Tan Y, Liu A, Yue X, Chen Y, Sangiovanni-Vincentelli A (2020) Using ensemble classifiers to detect incipient anomalies. arXiv: 2008.08710
22. Kaur G (2020) A comparison of two hybrid ensemble techniques for network anomaly detection in spark distributed environment. J Info Security Appl 55:102601. https://doi.org/10.1016/j.jisa.2020.102601
23. Ko AHR, Sabourin R, Britto AS Jr (2008) From dynamic classifier selection to dynamic ensemble selection. Pattern Recogn 41(5):1718–1731. https://doi.org/10.1016/j.patcog.2007.10.015
24. Krawczyk B, Minku LL, Gama J, Stefanowski J, Woniak M (2017) Ensemble learning for data stream analysis. Inf Fusion 37(C):132–156. https://doi.org/10.1016/j.inffus.2017.02.004
25. Kumar G, Thakur K, Ayyagari MR (2020) Mlesidss: machine learning-based ensembles for intrusion detection systems-a review. J Supercomput 76:8938–8971
26. Lipowski A, Lipowska D (2012) Roulette-wheel selection via stochastic acceptance. Physica A: Stat Mech Appl 391(6):2193–2196. https://doi.org/10.1016/j.physa.2011.12.004
27. Lv F, Han M, Qiu T (2017) Remote sensing image classification based on ensemble extreme learning machine with stacked autoencoder. IEEE Access 5:9021–9031
28. Margineantu DD, Dietterich TG (1997) Pruning adaptive boosting ICML 97:211–218
29. Nawaz A, Rehman AU, Abbas M (2020) A novel multiple ensemble learning models based on different datasets for software defect prediction. arXiv:2008.13114
30. Rezaei-Ravari M, Eftekhari M, Saberi-Movahed F (2021) Regularizing extreme learning machine by dual locally linear embedding manifold learning for training multi-label neural network classifiers, p 104062
31. Saini R, Ghosh SK (2017) Ensemble classifiers in remote sensing: A review. In: 2017 International Conference on Computing, Communication and Automation (ICCCA), pp 1148–1152
32. Siddiqui A, Boukerche A (2020) Tempocode-iot: temporal codebook-based encoding of flow features for intrusion detection in internet of things. Cluster Comput 24:17–35
33. Skurichina M, Duin RP (1998) Bagging for linear classifiers. Pattern Recogn 31(7):909–930. https://doi.org/10.1016/S0031-3203(97)00110-6
34. Soares RGF, Santana A, Canuto AMP, de Souto MCP (2006) Using accuracy and diversity to select classifiers to build ensembles. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp 1310–1316. https://doi.org/10.1109/IJCNN.2006.246844
35. Su HY, Lin YJ, Chu CC (2021) Applications of decision tree and random forest methods for real-time voltage stability assessment using wide area measurements. In: Wide Area Power Systems Stability, Protection, and Security. Springer, pp 373–391
36. Sundareswaran A, Lavanya K (2020) Real-time vehicle traffic prediction in apache spark using ensemble learning for deep neural networks. Int J Intell Info Tech (IJIIT) 16(4):19–36
37. Tang J, Su Q, Su B, Fong S, Cao W, Gong X (2020) Parallel ensemble learning of convolutional neural networks and local binary patterns for face recognition. Comput Methods Programs Biomed 197:105622. https://doi.org/10.1016/j.cmpb.2020.105622
38. Tewari S, Dwivedi U (2020) A comparative study of heterogeneous ensemble methods for the identification of geological lithofacies. J Petroleum Exploration Prod Tech 10:1849–1868
39. Woloszynski T, Kurzynski M, Podsiadlo P, Stachowiak GW (2012) A measure of competence based on random classification for dynamic ensemble selection. Info Fusion 13(3):207–213
40. Woods K, Kegelmeyer WP, Bowyer K (1997) Combination of multiple classifiers using local accuracy estimates. IEEE Trans Pattern Anal Mach Intell 19(4):405–410. https://doi.org/10.1109/34.588027
41. Zhang ML (2009) ML-RBF: RBF neural networks for multi-label learning. Neural Process Lett 29:61–74. https://doi.org/10.1007/s11063-009-9095-3
42. Zhang ZL, Chen YY, Li J, Luo XG (2019) A distance-based weighting framework for boosting the performance of dynamic ensemble selection. Info Process Manage 56(4):1300–1316. https://doi.org/10.1016/j.ipm.2019.03.009
43. Zhang T, Chi G (2020) A heterogeneous ensemble credit scoring model based on adaptive classifier selection: an application on imbalanced data. Int J Fin Econ. https://doi.org/10.1002/ijfe.2019
44. Zhang M-L, Zhou Z-H (2007) ML-KNN: a lazy learning approach to multi-label learning, pp 2038–2048

45. Zhao Y, Xue J, Chen X (2015) Ensemble learning approaches in speech recognition, pp 113–152. Springer. https://doi.org/10.1007/978-1-4939-1456-2_5
46. Zhou ZH, Zhang ML (2017) Multi-label Learning. Springer, US, Boston, MA, pp 875–881
47. Zvarevashe K, Olugbara O (2020) Ensemble learning of hybrid acoustic features for speech emotion recognition. Algorithms 13:70. https://doi.org/10.3390/a13030070