IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 29, 2021, accepted July 14, 2021, date of publication July 26, 2021, date of current version August 4, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3099689

Automatic Recommendation Method for Classifier Ensemble Structure Using Meta-Learning

ROBERCY ALVES DA SILVA[1], ANNE MAGÁLY DE PAULA CANUTO[1], CEPHAS ALVES DA SILVEIRA BARRETO[1], AND JOÃO CARLOS XAVIER-JUNIOR[2]

[1]Department of Informatics and Applied Mathematics, Federal University of Rio Grande do Norte, Natal 59078-970, Brazil
[2]Digital Metropolis Institute, Federal University of Rio Grande do Norte, Natal 59078-970, Brazil

Corresponding author: João Carlos Xavier-Junior (jcxavier@imd.ufrn.br)

ABSTRACT Machine Learning (ML) is a field that aims to develop efficient techniques to provide intelligent decision making solutions to complex real problems. Among the different ML structures, a classifier ensemble has been successfully applied to several classification domains. A classifier ensemble is composed of a set of classifiers (specialists) organized in a parallel way, and it is able to produce a combined decision for an input pattern (instance). Although Classifier ensembles have proved to be robust in several applications, an important issue is always brought to attention is the ensemble's structure. In other words, the correction definition of its structure, like the number and type of classifiers and the aggregation method, has an important role in its performance. Usually, an exhaustive testing and evaluation process is required to better define the ideal structure for an ensemble. Aiming to produce an interesting investigation in this field, this paper proposes two new approaches for automatic recommendation of classifier ensemble structure, using meta-learning to recommend three of these important parameters: type of classifier, number of base classifiers, and the aggregation method. The main aim is to provide a robust structure in a simple and fast way. In this analysis, five well known classification algorithms will be used as base classifiers of the ensemble: kNN (Nearest Neighbors), DT (Decision Tree), RF (Random Forest), NB (Naive Bayes) e LR (Logistic Regression). Additionally, the classifier ensembles will be evaluated using seven different strategies as aggregation functions: HV (Hard Voting), SV (Soft Voting), LR (Logistic Regression), SVM (Support Vector Machine), NB(Naive Bayes), MLP (Multilayer perceptron) e DT (Decision Tree). The empirical analysis shows that our approach can lead to robust classifier ensembles, for the majority of the analysed cases.

INDEX TERMS Classifier ensembles, meta-learning, multiple classifier system, machine learning.

I. INTRODUCTION

Classifier ensembles, also called ensembles, is a more elaborated Machine Learning structure that has been successfully applied in Pattern Recognition applications [1]–[3]. These systems are broadly used to solve a wide range of problems, such as face recognition [4], music classification [5], credit scoring [6], recommendation systems [7]–[9], software bug prediction [10], intruder detection [11], machine leaning, pattern recognition, knowledge discovery [12], and many other problems found in the real world.

The associate editor coordinating the review of this manuscript and approving it for publication was Jad Nasreddine.

However, defining the best components of an ensemble's structure is a complex task, since its performance is directly related to the characteristics of a particular problem. Moreover, components such as neural networks may have different results depending on their structure or their initialization parameters. Thus, the best configuration of an ensemble varies according to its application [13].

One of the alternatives for automating this definition process is through meta-learning. In this context, the main goal of meta-learning is to understand the interaction between the learning mechanism and the concrete contexts in which it is applicable [14]. This can be achieved by applying Machine Learning techniques to build models that explain the relationship between learning strategies and problems from a

particular perspective. Meta-learning explores accumulated knowledge about various tasks and their possible applications in finding solutions to problems that are similar to those that originated such knowledge.

In the literature, several studies that use meta-learning to recommend algorithms can be found, such as in [13], [15]–[17] and [18]. However, although there are several relevant studies, it is clear that there is a lack of investigation towards the possibility of using recommendation methods to assist in effectively defining the main parameters of an ensemble, notably the classifier type, the number of classifiers and the combination methods.

The main aim of this work is to propose an efficient system for automatic recommendation of the structure of classifier ensembles. In order to achieve this goal meta-learning is used to define the best configuration of parameters for this structure. The concept of meta-learning will be applied in recommending three ensemble-specific project parameters: 1) Base classifier: the best set of base classifiers for an ensemble; 2) Ensemble size: the most appropriate number of base classifiers that will form the ensemble; and 3) Combination method: the best model for combining the results of the various base classifiers. To the best of our knowledge, there is no work that use meta-learning to recommend a set of ensemble parameters. Usually, they recommend only one ensemble parameter. In addition, very little has been done to recommend the aggregation function of a classifier ensemble.

In order to assess the feasibility of the proposed automatic recommendation system, an empirical analysis will be conducted, assessing the performance of the proposed system using 100 datasets. Additionally, a comparative analysis will also be performed, comparing the obtained results of the proposed system with some well-known ensemble-based systems.

This work is organized as follows: Section II presents the basic concepts of this paper while Section III presents the state-of-the-art technologies and the main advances on the subject of ensemble recommendation systems. In Section IV, we propose a method that uses a meta-learning concept for the recommendation of the size and type of classifiers and aggregation model of an ensemble of classifiers. Section V will describe the experimental methodology that is used in the empirical analysis, while the reported results are presented and analyzed in Section VI. Finally, the conclusion and perspectives for future researches are presented in Section VII of this work.

II. THEORETICAL REFERENCE

This section aims to present the main theoretical foundations that are used during the conception of this paper. Therefore, the next two subsection will present an explanation about classifier ensembles and meta-learning, respectively.

A. CLASSIFIER ENSEMBLE

It is well-known that there is not a single classifier which can be considered optimal for all problem domains. Therefore, it is difficult to find a good single classifier which provides the best performance in practical pattern classification tasks [3], [12].

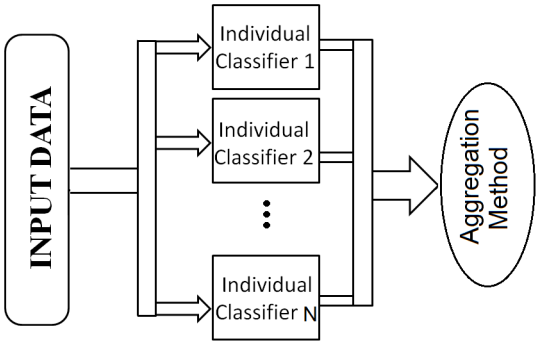

FIGURE 1. The general structure of an ensemble system.

Figure 1 presents a general structure of an ensemble, which consists of a set of c individual classifiers (ICs) and an aggregation module ($Comb$). Therefore, an input pattern $\{x_i \in R^d | i = 1, 2, \ldots, n\}$ is presented to all individual classifiers, and an aggregation method will combine their outputs to produce the overall output of the system $O = Comb(y_j)$, $\{y_j = (y_{j1}, \ldots, y_{jk} | j = 1, \ldots, c \text{ and } k = 1, \ldots, l\}$, where the number of individual classifiers is defined by c, and l describes the number of labels in a dataset.

One important issue regarding the design of classifier ensembles involves the appropriate selection of number and type of individual classifiers, and also the combination function (aggregation method). Machine Learning literature has ensured that diversity plays an important role in the design of ensembles, contributing to their accuracy and generalization [19]. The ideal situation would be a set of classifiers that present uncorrelated errors, also called diversity. In this paper, the composition of an ensemble is defined by a meta-learning recommendation system, that will be described in the next subsection.

B. META-LEARNING RECOMMENDATION SYSTEM

Recently, meta-learning techniques have emerged as an efficient alternative for ensemble parameters recommendation [20], [21]. The idea of meta-learning as a recommendation system can be applied to individual classifiers or to ensembles. In the case of ensembles, the performance of an ensemble is related to a set of characteristics (meta-features) of the corresponding problem. Hence, it acquires knowledge based on the parameters of each configuration of the ensemble system. Then, this acquired knowledge is used in the design of an ensemble, when a new task is presented. According to [16], in general, the design of an ensemble recommendation system is composed of four main steps, which are:

1) Dataset characterization: in this step, the main meta-features need to be discovered, so that they can be applied to the meta-learner. One of the first studies to extract meta-features from a specific dataset was presented by the Statlog project [22].

2) Definition of evaluation metrics: here, the process of selecting the best algorithm is performed. In this case, it is necessary to apply evaluation measures in order to select the best model to solve a specific problem, taking into account the more satisfactory performance for the analyzed problem. In this step, several evaluation metrics can be employed to measure the effectiveness of the used algorithms. In this paper, we will use accuracy as the main evaluation metric;
3) Definition of the recommendation output: the third step is related to the final result that will be presented by the recommendation system. The authors in [16] suggest three techniques: 1) definition of the best algorithm; 2) definition of a group of best algorithms; or 3) a ranking of the best algorithms. In this paper, we select and recommend a group of best algorithms;
4) Development of the recommendation model: here, the goal is to learn an implicit mapping between meta-features and classes in the meta-label.

The set of available meta-instances is called metadata. In order to induce the mapping between input meta-features and meta-class, a machine learning algorithm, which is called meta-learner, is applied. Through it, it is possible to generate the recommendation of the ensemble structure. Initially, an ensemble size recommendation will be made. After that, the base classifier recommendation will be made, and finally the aggregation method will be defined.

III. RELATED WORK

Classifier ensembles have been proved to be an efficient pattern recognition structure that has been applied to different applications [23], [24]. Despite the large number of researches on classifier ensembles, finding an optimal parameter set that maximizes classification accuracy of an ensemble is still an open problem. The search space for all parameters of an ensemble system (type of classifier, size, classifier parameters, combination method and feature selection) is very large and the definition of the optimal parameter set is a hard research challenge.

In Machine Learning, the choice of the best technique for a particular classification problem is a challenge which has been addressed by several authors [25]–[31]. This problem has been treated as a meta-learning problem [27], [32], automatic selection of machine learning (auto-ML) [26], [28], [30] or an optimization problem [25], [29], [31].

A considerable amount of meta-learning research has been devoted to the area of algorithm recommendation. In this special case of meta-learning, the aspect of interest is the relationship between data characteristics and algorithm performance, with the final goal of predicting an algorithm or a set of algorithms suitable for a specific problem under study. This application of meta-learning can be both useful for providing a recommendation to an end-user and for automatically selecting or weighting algorithms that are most promising to a specific problem.

The first work to present an abstract model for meta-learning is in [33], whereas the authors in [34] developed rules based on simple meta-features only determining whether a certain algorithm should be used for a problem instance or not. This approach was later extended by using more features and a Decision Tree learner based on StatLog project [35].

The authors in [36] presented a new strategy to recommend algorithms by using the K-NN algorithm to identify the most similar historical datasets. In [37], the authors tried to find functions that map datasets to algorithm performance. In [38], the problem complexity measures to characterize the datasets was used and the relation between these measures and performance of classification algorithms was analyzed. In [39], the authors proposed a rule-based classifier selection approach based on the technique proposed in [40], [41]. In the mentioned work, not only the algorithms themselves were recommended, but different parameter settings that will naturally led to performance variation of the same algorithm on different datasets.

As more recent studies, we can cite [42], in which a study was conducted for the construction of a symbolic recommendation model of the best Feature Selection algorithm. In addition, a lazy method was presented in [43] for the recommendation of Feature Selection algorithms, while a meta-learning framework was developed in [44] to learn which Feature Selection algorithms are more suitable for a given dataset. The authors in [45] used five different categories of state-of-the-art meta-features to characterize datasets, and built a different regression model to connect datasets to each candidate algorithm. In [46], a new approach for meta-feature engineering was introduced. Finally, in [47], the authors developed a method that based on a ranking list, determines which aggregation algorithms are best for that list.

In summary, the majority of the researches use meta-learning to recommend the best algorithm and/or parameter for a single classifier. To the best of our knowledge, there is no work that recommend the whole ensemble structure (classifier type, size and aggregation functions) using meta-learning.

IV. ENSEMBLE RECOMMENDATION SYSTEMS

In this Section, we will present two approaches for Ensemble Recommendation Method (ERM) proposed in this paper. The general architecture and operations will be presented, demonstrating the main steps that involve the proposed process of recommending the best topology of an ensemble using meta-learning.

The two proposed approaches for ERM are named as follows: 1) **ERM-ML** - *Ensemble Recommendation Method - Using Meta-learning*; and 2) **ERM-3ML** - *Ensemble Recommendation Method - Using 3 steps Meta-learning* .

A. RECOMMENDATION OF ENSEMBLE PARAMETERS

It is well known that different ensemble configurations lead to different performance results, and the use of meta-learning may be an excellent option to help in selecting the best

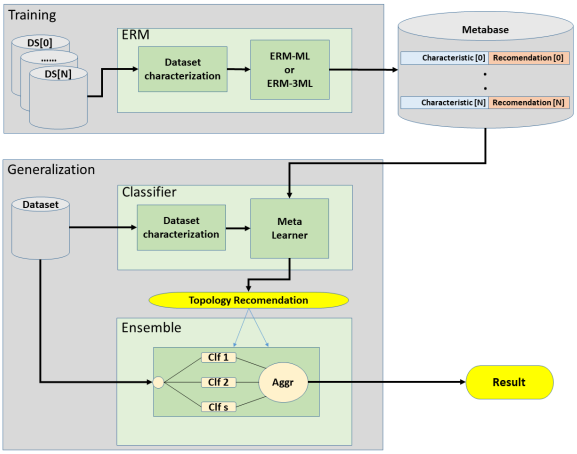

FIGURE 2. Flowchart of the proposed method.

ensemble parameter set for a specific problem. Since the definition of the ensemble structure is a crucial step in its project, the proposal of this work will be based on the development of two models to recommend this structure using meta-learning techniques. In general terms, the flowchart of the proposal of this work is shown in Figure 2. In this flowchart two very distinct phases are highlighted:

1) *Training:* This step is responsible for building the Meta-base. The more datasets are evaluated, the greater the expectation of performance and generalization of the recommendation system. This Meta-base will have the meta-features of all used datasets as well as the recommendation of the best structure for each instance (meta-features of a problem);
2) *Generalization:* This step is responsible for creating, training and applying the meta-leaner in real world problems, providing the best ensemble structure made by the proposed models. This step will extract the meta-features from new datasets (instances), and use a simple classification model (meta-learner) to recommend the best ensemble structure.

In the training and generalization steps, the built meta-feature database will have a structure similar to the one shown in Figure 3. Once the Meta-base is assembled, the original datasets will no longer be required, since all processing will be done only on the Meta-base, considerably reducing the computational effort of the proposed model.

In the generalization step, the Meta-learner method will be used to provided the recommendation of the best structure (number of classifiers, type of classifier and model aggregation) for the proposed ensemble.

B. ERM-ML- ENSEMBLE RECOMMENDATION METHOD - USING META-LEARNING

The ERM-ML method was first proposed in [48]. As it can be seen in Figure 4, the basic idea of this method is to initially train several algorithms using different sizes of classifiers and

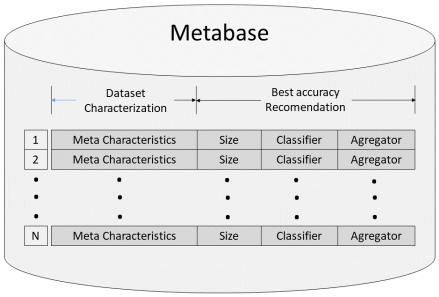

FIGURE 3. Structure for the meta-base.

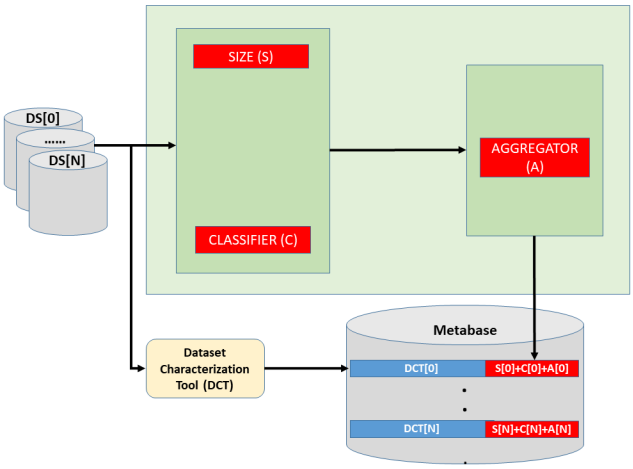

FIGURE 4. General structure of the ERM-ML method.

aggregation methods. Based on this result, it is possible to obtain the best ensemble structure for a given classification dataset. Subsequently, the obtained meta-features are used to train a classifier (meta-leaner) that will be tested unseen classification problem data.

In order to better understand this proposed method, suppose that DS is a dataset, consisting of $A = \{att_1, att_2, \ldots, att_d\}$ attributes, and N instances, where d is the total number of attributes of DS. The last attribute of this dataset is the class label of the instance. The instances will be divided into 2 sets: training $TR = \{tr_1, tr_2, \ldots, tr_{nt}\}$ and validation $V = \{v_1, v_2, \ldots, v_{nv}\}$. Where nt and nv represent the sizes of the training and validation sets, respectively. Algorithm 1 presents the main steps employed by the proposed method to create the Meta-base.

The steps of the proposed method in Algorithm 1 must be executed for each dataset, and they can be described as follows:

1) Lines 1 to 5 set a dataset DS to be trained and they define the possible recommendations for the main parameters of the ensemble structure: poolSize (number of classifiers); poolClassifier (number of classification models); and poolAggregator (aggregation functions). Additionally, by using the DCT tool,

Algorithm 1 Algorithm to Create ERML-ML Meta-Base

Input: dataset DS.

1: Open Meta-base file: MB
2: poolSize ← Vector with ensemble sizes
3: poolClassifier ← Vector with classification models
4: poolAggregator ← Vector with aggregation functions
5: DCT ← dataset_characterization_tool(DS)
6: **for** each size in poolSize **do**
7: **for** each model in poolClassifier **do**
8: **for** each aggregator in poolAggregator **do**
9: Result[i] ← Ensemble(DS, size, model, aggregator)
10: $S[i]$ ← size
11: $C[i]$ ← model
12: $A[i]$ ← aggregator
13: $i \leftarrow i + 1$
14: **end for**
15: **end for**
16: **end for**
17: Best ← SelectBest(max(Result[]))
18: MB ← Concatenate (MB, DCT, (S[Best] + C[Best] + A[Best]))

the desired characteristics of the dataset *DS* are extracted.

2) In this nested loop, from lines 6 to 16, a classifier ensemble is applied, varying its size, classifier types and aggregation function. Then, the selected model is trained with the defined training dataset, and validated with the validation dataset. This process is applied for all possible combination of size, classifier type and aggregation function. Each accuracy result, for each combination, will be stored in the *Result* vector. The current size, classifier and aggregation function will be also stored in vectors S, C and A, respectively;
3) After the evaluation loop (from line 17 onward), the selection of the best topology is performed by selecting the best ensemble configuration using the *SelectBest* procedure. In this procedure, one parameter is fixed and we calculate the average accuracy for each value of this parameter using all possibilities of the remaining parameters. Then, the value that contains the highest average accuracy is selected. For instance, in order to define the best classification model, we calculate the average accuracy for each classification model using all possibilities of ensemble sizes and aggregation functions of this classification model. We do the same idea to select the best ensemble size and aggregation function independently of the already selected parameters. Once these values are defined, the DCT information and the best values ($S[Best]$, $C[Best]$ and $A[Best]$) are stored in the Meta-base file *MB*, that will be available to be used with new test examples in the Application step.

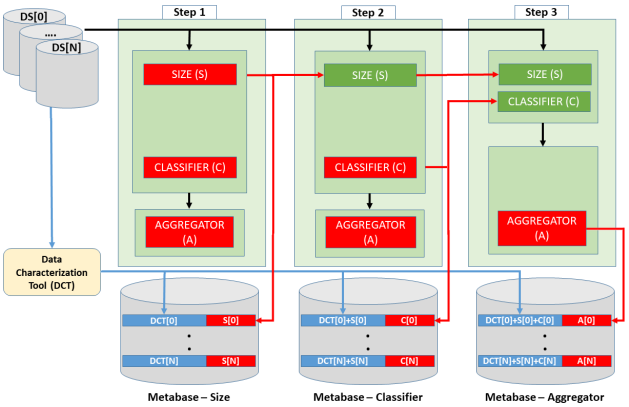

FIGURE 5. General structure of the ERM-3ML method.

C. ERM-3ML - ENSEMBLE RECOMMENDATION METHOD - USING 3 STEPS META-LEARNING

The need of analyzing the behavior of error propagation during the recommendation of each of the three parameters has inspired the development of this proposed method. Figure 5 shows the block diagram of the recommendation proposal of the ERM-3ML (Ensemble Recommendation Method - Using 3 steps Meta-learning). The basic idea of this approach is, initially, to train sequentially meta classifiers to the ensemble size, then the classification algorithms and, finally, the aggregation function. Then, from these training steps, it is possible to obtain the best ensemble structure for a given dataset. Therefore, the main difference between ERM-3ML and ERM-ML is that ERM-ML selects each ensemble parameter in an independent and parallel way while ERM-3ML performs a serial recommendation. In order to do this, ERM-3ML creates three datasets, in a serial way, in which the output of one meta classifier is used as input attribute for the following meta classifier.

In order to better understand this proposed version, suppose *DS* is a dataset composed by $A = \{att_1, att_2, \ldots, att_d\}$ attributes and N instances, in which d is the total number of attributes of *DS*. The instances will be divided into 2 sets: training $TR = \{tr_1, tr_2, \ldots, tr_{nt}\}$ and validation $V = \{v_1, v_2, \ldots, v_{nv}\}$. Where n and nv represent the sizes of the training and validation sets, respectively. Algorithm 2 presents the main steps used by the proposed method, in the Training and Validation phase.

The steps of the proposed method in Algorithm 2 must be executed for each new dataset, and can be described as:

1) Lines 1 to 7 set a dataset *DS* to be trained, and they define the possible recommendations for the main parameters of the ensemble structure: poolSize (number of classifiers); poolClassifier (number of classification models); and poolAggregator (aggregation functions). In addition, by using the DCT tool, we extract the desired characteristics from the dataset *DS*.
2) In the first nested loop from lines 8 to 16, a classifier ensemble is evaluated, sequentially, varying size, classifier and aggregation functions. This process is

Algorithm 2 Algorithm to Create ERML-3ML Meta-Bases

Input: dataset DS.

```
1: Open size Meta-base file: MBS
2: Open Classifier Meta-base file: MBC
3: Open aggregator Meta-base file: MBA
4: poolSize ← Vector with ensemble sizes
5: poolClassifier ← Vector with classification models
6: poolAggregator ← Vector with aggregation functions
7: DCT ← dataset_characterization_tool(DS)
   {creation of the meta-base_size}
8: for each size in poolSize do
9:    for each model in poolClassifier do
10:      for each aggregator in poolAggregator do
11:         Result[i] ← Ensemble(DS, size, model, aggrega-
            tor)
12:         S[i] ← size
13:         i ← i + 1
14:      end for
15:   end for
16: end for
17: BestSize ← S[index(max(Result[])])
18: MBS ← Concatenate (MBS, DCT, BestSize)
    {creation of the meta-base_classifier}
19: for each model in poolClassifier do
20:   for each aggregator in poolAggregator do
21:      Result[i] ← Ensemble(DS, BestSize, model, aggre-
         gator)
22:      C[i] ← model
23:      i ← i + 1
24:   end for
25: end for
26: BestClassifier ← C[index(max(Result[])])
27: MBC ← Concatenate (MBC, DCT, BestSize, BestClas-
    sifier)
    {creation of the meta-base_aggregator}
28: for each aggregator in pool_Aggregator do
29:   Result[i] ← Ensemble(DS, BestSize, BestClassifier,
      aggregator)
30:   A[i] ← aggregator
31:   i ← i + 1
32: end for
33: BestAggregator ← A[index(max(Result[])])
34: MBA ← Concatenate (MBA, DCT, BestSize, BestClas-
    sifier, BestAggregator)
```

applied for all possible combination of size, classifier and aggregation function. Each accuracy result, for each combination, will be stored in the vector *Result*.

3) In lines 17 and 18, the selection of the first parameter is performed by selecting the number of classifiers with the best accuracy value presented in the vector *Result*, and stores it in the size Meta-base file *MBS*;
4) In the second nested loop, from line 19 to 25, a classifier ensemble is applied, using the best number of classifiers selected in previous item, varying classifier types and aggregation functions. Then, in lines 26 and 27, we select the second parameter, applying a function to select the type of classifier with the best accuracy value presented in the vector *Result*, and store it in the classifier Meta-base file *MBC*;
5) In the third nested loop from lines 28 to 32, an ensemble classifier, using the best number and types of classifiers are applied, varying the aggregation function. Then, in line 33, we select the third parameter, applying a function to select the aggregation function with the best accuracy value present in the vector *Result*, and store it in the aggregator Meta-base file *MBA*. Meta-bases *MBA*, *MBC* and *MBS* will be available for use with new test instances in the Application phase.

The main difference between both algorithms is that the former creates only one mate-base with the recommendation of all three parameters, while the latter creates three meta-bases, in a incremental way, one for each parameter.

V. EXPERIMENTAL PROTOCOL

In order to evaluate the design of our recommendation framework, we present an empirical comparison using 8 state-of-the-art techniques and our two recommendation methods under the same experimental protocol. All algorithms used in this work were implemented in Python Programming Language (sklearn package). In this section, we provide a complete description of the experimental analysis of this paper.

A. DATASETS

This experimental analysis is performed using a test bed composed of 100 datasets, of which 50 datasets were taken from the UCI machine learning repository [49], from the Statlog project [35] and from the Knowledge Extraction based on Evolutionary Learning (KEEL) repository [50]. The remaining 50 datasets were artificially generated from other datasets, using the SMOTE function ($SMOTE-C0-K5-P100-S1$), available in WEKA[1] software. The key characteristics of such datasets can be found in Table 1.

B. COMPARISON METHODS

Given that the proposed methods are characterized as ensemble system, in order to assess their effectiveness, they will be compared with the most relevant Dynamic Selection algorithms presented by [51], and which are listed in Table 2.[2] Among the selected methods, three different approaches are used: Dynamic Classifier Selection (DCS), Dynamic Ensemble Selection (DES) and Topology Recommendation System (TRS).

The obtained results of all analysed methods will be evaluated using the Friedman statistical test [58]. In cases where a statistically significant difference is detected, the Nemenyi post-hoc test is applied [58]. In order to present the obtained

[1] https://www.cs.waikato.ac.nz/~ml/weka/

[2] All Dynamic Selection algorithms were taken from https://github.com/scikit-learn-contrib/DESlib

TABLE 1. Main characteristics of the datasets used in the experiments.

Datasets	# Inst.	# Attrib.	# Classes
iris, iris0, iris1, iris2, iris3, iris4, iris5, iris6, iris7, iris8, iris9 and irisa	150	4	3
glass, glass1, glass2, glass3, glass4, glass5, glass6, glass7, glass8, glass9 and glassa	214	9	7
sonar	208	60	2
haberman	306	3	2
ionosphere	351	34	2
liver-disorders	345	6	2
heart-statlog	270	13	2
diabetes, diabetes1, diabetes2, diabetes3, diabetes4, diabetes5, diabetes6, diabetes7, diabetes8, diabetes9 and diabetesa	768	8	2
balance-scale	625	4	3
bupa, bupa1, bupa2, bupa3, bupa4, bupa5, bupa6, bupa7, bupa8, bupa9 and bupaa	345	6	2
hill-valley	606	100	2
vehicle	846	18	4
waveform-5000	5,000	40	3
pendigits	10,992	16	10
letter	20,000	16	26
kdd-synthetic-control	600	61	6
mfeat-factors	2,000	74	10
monkes-3	122	6	2
page-blocks	5,473	10	5
tae	151	5	3
data-banknote-authentication	1,372	4	2
fertility	100	9	2
seeds	210	7	3
balance-scale	625	4	3
segment	2,310	19	7
transfusion	748	4	2
phishing	1,353	9	3
yeast	1,484	8	10
wifi-localization	2,000	7	4
wall-following	5,456	4	4
vowel-context	990	13	11
spect	80	22	2
column	310	6	3
magic-04	19,020	10	2
wholesale-region	440	6	3
ecoli, ecoli1, ecoli2, ecoli3, ecoli4, ecoli5, ecoli6, ecoli7, ecoli8, ecoli9 and ecolia	332	7	6
cryotherapy	90	6	2
breast-tissue	106	9	6
wdbc	569	30	2
cardiotocography	2,126	19	3
mammographic-m	961	5	2
mfeat-factors-AttributeSelection	2,000	74	10
mfeat-pixel-AttributeSelection	2,000	103	10
mfeat-fourier-AttributeSelection	2,000	38	10
mfeat-morphological	2,000	6	10
mfeat-zernike	2,000	47	10
mfeat-pixel	2,000	240	10
optdigits	5,620	64	10
mfeat-karhunen	2,000	64	10
mfeat-fourier	2,000	76	10

TABLE 2. Dynamic ensemble methods.

Selection Approach	Technique	Selection Criteria	Reference	Year
DCS	Classifier Rank (rank)	Ranking	Sabourin et al. [52]	1993
	Overall Local Accuracy (OLA)	Accuracy	Woods et al. [53]	1997
	Local class accuracy (LCA)	Accuracy	Woods et al. [53]	1997
	Multiple Classifier Behavior (MCB)	Behavior	Giacinto et al. [54]	2001
DES	K-Nearest Oracles Eliminate (KNE)	Oracle	Ko et al. [55]	2008
	K-Nearest Oracles Union (KNU)	Oracle	Ko et al. [55]	2008
	DES Performance (DESP)	Probabilistic	Woloszynski et al. [56]	2012
	META-DES (META)	Meta-learning	Cruz et al. [57]	2015

results by the post-hoc test, the critical difference diagram is used. This diagram was selected in order to have a visual illustration of the statistical test, making it easier to interpret the obtained results.

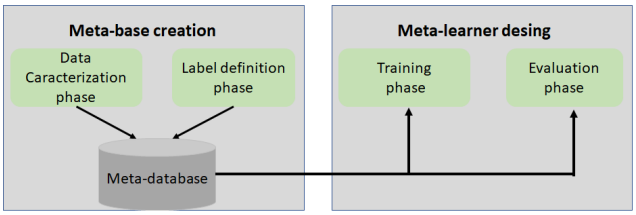

FIGURE 6. The experimental methodology.

C. EXPERIMENTAL METHODOLOGY

The experimental methodology used in this empirical (Figure 6) analysis is divided into four phases: 1) data characterization phase; (2) meta-base label definition phase; (3) meta-learner training phase; and (4) meta-learner evaluation phase. The first two phases are related to the creation of the meta-base whereas the remaining two phases are related to the training and evaluation of the meta-learner.

1) META-BASE DATA CHARACTERIZATION PHASE

In order to create a meta-base, its attributes have to defined and this is done in the data characterization phase. Before the data characterization phase, the original datasets goes through a pre-processing phase, filling missing values and normalizing all numeric attributes.

According to [13], the measures that characterize the databases must contain relevant information to determine the relative performance among classification algorithms, and present low computational cost. Currently, the dataset characterization research focuses on three main aspects [59]: direct characterization, characterization based on landmarking, and characterization via models. Here we decided to adopt the direct characterization, based on the Statlog project [22]. More recently, the METAL project has been proposed[3] aiming at developing tools to assist the user in selecting an appropriate combination of pre-processing, classification and regression techniques. Table 3 displays all 25 meta-features considered in this paper, extracted by direct characterization of the datasets, using the DCT (Data Characterization Tool), proposed by the METAL project.

2) META-BASE LABEL DEFINITION PHASE

Once the attributes of the meta-base are defined, the next step is the definition of the labels for the instances of this meta-base. In order to do that, a brute force approach is performed, in which each instance (classification problem) will be submitted to a set of possible combinations of type of classifiers, number of classifiers and type of aggregation functions. After this, the best ensemble configuration is selected and put as label of the corresponding instance.

In relation to the number of classifiers in an ensemble, the values vary as follows: $PoolSize = [2, 5, 8, 10, 12, 15, 18, 20, 22, 25, 28, 30, 32, 35, 38, 40, 42, 45, 48, 50]$. Moreover,

[3]http://www.metal-kdd.org

TABLE 3. The meta-features obtained from datasets by the DCT tool.

DCT Meta-attribute	Description
Nr_attributes	Number of attributes
Nr_sym_attributes	Number of symbolic attributes
Nr_num_attributes	Number of numerical attributes
Nr_examples	Number of records/examples
Nr_classes	Number of classes
Default_accuracy	The default accuracy
MissingValues_Total	Total number of missing values
Lines_with_MissingValues_Total	Number of examples having missing values
MeanSkew	Mean skewness of the numerical attributes
MeanKurtosis	Mean kurtosis of the numerical attributes
NumAttrsWithOutliers	Number of attributes for which the ratio between alpha-trimmed and standard deviation, and standard deviation is larger than 0.7
MStatistic	Boxian M-Statistic to test for equality of covariance matrices of the numerical attributes
MStatDF	Degrees of freedom of the M-Statistic
MStatChiSq	Value of the Chi-Squared distribution
SDRatio	A transformation of the M-Statistic which assesses the information in the co-variance structure of the classes
Fract	Relative proportion of the total discrimination power of the first discriminant function
Cancor	Canonical correlation of the best linear combination of attributes to distinguish between classes
WilksLambda	Discrimination power between the classes
BartlettStatistic	Bartlett V-Statistic to test the significance of the discriminant functions
ClassEntropy	Entropy of classes
EntropyAttributes	Average entropy of symbolic attributes
MutualInformation	Mutual information between symbolic attributes and classes
JointEntropy	Average joint entropy of the symbolic attributes and the classes
Equivalent_nr_of_attrs	Ratio between class entropy and average mutual information
NoiseSignalRatio	Ratio between noise and signal, indicating the amount of irrelevant information for classification

the PoolClassifier vector will contain five well known classification algorithms, which are: *k*-NN (Nearest Neighbors), DT (Decision Tree), RF (Random Forest), NB (Naive Bayes) e LR (Logistic Regression). These classification algorithms were selected due to the different learning criteria that they provide. In addition, they have been widely used in many application domains. These algorithms are trained in a Bagging-based procedure in a 10-fold cross validation process. It is important to emphasize that the training process is made using the original classification dataset. Additionally, the classifier ensembles use up to seven different strategies as aggregation functions: HV (Hard Voting), SV (Soft Voting), LR (Logistic Regression), SVM (Support Vector Machine), NB(Naive Bayes), MLP (Multilayer perceptron) e DT (Decision Tree).

As five classification models are used in the definition of the meta-base label, the base classifier is selected using the following procedure.

1) For each classification model, several configuration (hyper-parameters) are assessed and the average is calculated.
2) The classification model with the highest average accuracy is selected.

The recommended ensemble has a homogeneous structure and the configurations with the highest accuracies are selected to be part of the ensemble.

As all the aforementioned classifiers have hyper-parameters, for each base classifier, a hyper-parameter is randomly selected from a pre-defined interval. Additionally, we perform 10 executions for each ensemble configuration (size, classifier and aggregation functions). For instance, in an ensemble composed of 10 *k*-NN classifiers, the first classifier is selected by choosing k from a $[2, 20]$ interval. The same procedure is performed to select the k-NN hyper-parameter of the following 9 base classifiers. Then, this ensemble is tested using all seven aggregation functions cited above. Finally, when recommending the base classifiers, the values obtained by all 70 executions (10 executions of 7 aggregation functions) are averaged. On the other hand, when recommending the aggregation function for a size, the values of this particular function with all five classifiers (10 executions of 5 base classifiers) are averaged to represent the averaged accuracy of this function. The main hyper-parameters are defined as following:

- *k-NN:* k lies in the $[2, 20]$ interval;
- *DT:* max_depth varies from 3 to 7 and pruning might be on or off;
- *RF:* Number of trees varies from 10 to 200 and max_depth varies from 3 to 7;
- *NB:* Numeric attributes might be treated as normal or kernel distribution;
- *LR:* C (inverse of regularization strength) varies from 0.1 to 2.0.

The other parameters of the classifiers were set to the defaults values of the Python language (sklearn package). For the aggregation functions, the following parameters are used.

- *LR:* The default values of the Python Language;
- *SVM:* The default values of the Python Language;
- *NB:* Default values of the Python Language;
- *MLP:* Number of hidden neurons is set to be the number of attributes plus the number of classes divided by 2 (average value of these two parameters);
- *DT:* The default values of the Python Language.

In case of draw, the selection of the best model is based on the first best overall accuracy, which means the smallest possible ensemble.

ERM-ML recommends the best size/classifier/aggregation parameter set in an independent way. This number was defined by the average of the best results obtained by all possibilities of the other parameters. For each instance (classification dataset), the best accuracy records will have the ensemble size,classification model and the type of aggregation function stored together with the characterization of the dataset in the Meta-base. At the end of the training, we will have as many instances as the number of datasets. Each instance will have x attributes, the first ones referring to the characteristics of the database, and the last representing the class with the best structure (size, classifier and aggregation).

Unlike ERM-ML, ERM-3ML will recommend all three parameters separately: size, classifier and aggregation function. In its first stage, ERM-3ML recommends the best size for that determined data profile. In the second stage, it uses the previous size information to recommend the best classifier for that particular size. Finally, with the previous size and classifier, it recommends the best aggregation function. In other words, the output of one meta classifier is used as input attribute for the following meta classifier. In this method, we will have three dataset, one for each parameter, and this phase defines the labels of all three datasets.

3) META-LEARNER TRAINING PHASE

Once the Meta-base is built, the next step is the creation of the meta-learner and it needs to be defined, trained and tested. For the selection of the meta-leaner model, an initial investigation was performed, in which the performance of four well-known classification algorithms were assessed, k-NN, SVM, MLP and C4.5. After this evaluation, SVM provided the best overall performance and it has been selected to be the meta-learner for both approaches.

As already mentioned, the meta-learner has been implemented using the Python Programming Language (sklearn package). The hyper-parameters were defined for each approach and are the following ones.

1) *ERM-ML:* SVM with a regularization parameter (C) set to 1.5, using a RBF kernel and kernel coefficient (gamma) set to scale. All other parameters were set to the Python default values;
2) *ERM-3ML-Size:* SVM with a regularization parameter (C) set to 2.1, using a Polynomial kernel and kernel coefficient (gamma) set to scale. All other parameters were set to the Python default values;
3) *ERM-3ML-Classifier:* SVM with a regularization parameter (C) set to 0.8, using a Sigmoid kernel and kernel coefficient (gamma) set to auto. All other parameters were set to the Python default values;
4) *ERM-3ML-Aggregation:* SVM with a regularization parameter (C) set to 1.1, using a Poly kernel and kernel coefficient (gamma) set to auto. All other parameters were set to the Python default values;

For training the meta-leaner, a 10-fold cross-validation method is applied.

4) META-LEANER EVALUATION PHASE

The last phase is the evaluation of the meta-learner. As already mentioned, in the case of ERM-ML, only one classifier will be used, which will be responsible for recommending, in a single step, the best size, classifier and aggregation function. On the other hand, in ERM-3ML, three classifiers and each classifier will be responsible for recommending one of three parameters of the topology at a time: size, classifier and aggregation function.

In order to evaluate the meta-learners, the methods trained in the previous phase are assessed. They can be assessed in two different ways: the accuracy of the meta-learner or the efficiency of the recommended ensemble. In this paper, we will evaluate the meta-learner based on the second way, the efficiency of the recommended ensemble. In this sense, once the meta-leaner defines the output of a testing instance, the recommended ensemble is created, trained and assessed (using the original dataset) in order to analyse its performance in the corresponding classification problem. The results of this analysis are presented in the next section.

VI. RESULTS AND DISCUSSION

In this section, the obtained results of the reference methods and the two proposed ones will be presented and analyzed, aiming to bring an interesting discussion on the overall results.

As mentioned previously, the meta-learner recommends the optimal ensemble structure (type and number of base classifiers and aggregation method) for a classification problem (testing instance). Then, the recommended ensemble is created, trained and assessed. Values presented in this section represent the accuracy levels provided by the recommended ensembles. For the reference methods, as we are dealing with a different classification problem, the whole process has to be done according to the algorithms defined by these methods.

A. PERFORMANCE OF ALL ANALYZED MODELS

The accuracy results of all ten analyzed methods are summarized in Tables 4 and 5. The results of this tables represent the accuracy level of the recommended ensemble structure for the corresponding dataset. For ERM-ML and ERM-3ML models, for each test instance (classification problem), a meta classifier recommends the best ensemble configuration. Then, the recommended ensemble is trained using the original dataset in a 10-fold cross validation methodology. Regarding the results presented in Tables 4 and 5, they represent the accuracy of the recommended ensemble over the original dataset. For the ensemble-based methods, the ensemble systems delivered by these methods are also assessed in a 10-fold cross validation procedure and the presented results are also the obtained accuracy levels. In this table, each column represents one analyzed method, highlighting in bold the best accuracy result (highest value) of each dataset. In order to provide a concise analysis, the final row summarizes the number of times each method delivered the best accuracy result.

As it can be seen in the two previous tables, the results are very promising, which are summarized in Table 6. It is worth noting that the results of our proposed methods are superior to the others for the majority of datasets. The best overall result is obtained by the ERM-3ML model, which achieved the best accuracy in 64 databases (35 bases in Table 4 + 29 bases in Table 5), followed by ERM-ML model in 47 databases (28 + 19). The best performance of an existing ensemble

TABLE 4. Results of method accuracy applied to original databases. List of competing methods: Multiple Classifier Behavior (MCB) [54], Overall Local Accuracy (OLA) [53], DES Performance (DESP) [56], K-Nearest Oracles Union (KNU) [55], Local class accuracy (LCA) [53], Classifier Rank (rank) [52], META-DES (META) [57], and our two methods: Ensemble Recommendation Method using Meta-Learning (ERMML) and Ensemble Recommendation Method using 3 steps Meta-Learning (ERM3ML).

	Methods									
Datasets	**MCB**	**OLA**	**DESP**	**KNU**	**LCA**	**RANK**	**KNE**	**META**	**ERMML**	**ERM3ML**
iris	0.940	0.960	0.960	0.960	0.960	0.960	0.960	0.960	**0.980**	**0.980**
glass	0.521	0.662	0.620	0.662	0.577	0.549	0.620	0.620	**0.746**	0.648
sonar	0.594	0.667	0.681	0.696	0.536	0.667	0.710	0.696	**0.768**	**0.768**
haberman	0.604	0.594	0.634	0.644	0.564	0.653	0.634	0.624	0.644	**0.673**
ionosphere	0.871	0.776	0.897	0.897	0.647	0.802	0.853	0.897	0.828	**0.922**
liver-disorders	0.579	0.588	0.658	0.649	0.526	0.640	0.632	0.649	**0.702**	0.693
heart-statlog	0.700	0.711	0.667	0.656	0.600	0.689	0.644	0.700	**0.789**	0.567
diabetes	0.701	0.728	0.720	0.736	0.744	0.709	0.713	0.673	0.756	**0.760**
balance-scale	0.768	0.754	0.787	0.802	0.763	0.768	0.744	0.773	**0.865**	**0.865**
bupa	0.579	0.588	0.658	0.649	0.526	0.640	0.632	0.649	0.684	**0.693**
hill-valley	0.445	0.480	0.465	0.420	0.505	0.420	0.435	0.465	**0.960**	**0.960**
vehicle	0.686	0.639	0.661	0.668	0.564	0.646	0.643	0.675	0.664	**0.789**
waveform-5000	0.711	0.720	0.793	0.795	0.733	0.719	0.724	0.764	**0.868**	**0.868**
pendigits	0.931	0.932	0.964	0.967	0.963	0.937	0.962	0.968	**0.989**	**0.989**
letter	0.785	0.776	0.839	0.849	0.805	0.792	0.820	0.842	**0.917**	**0.917**
kdd-synthetic-control	0.995	0.980	0.995	0.995	0.990	0.980	**1.000**	0.995	**1.000**	**1.000**
mfeat-factors	0.829	0.821	0.885	0.885	0.874	0.821	0.865	0.888	**0.927**	**0.927**
monkes-3	0.854	**0.902**	0.756	0.805	0.683	**0.902**	**0.902**	0.854	**0.902**	0.878
page-blocks	0.952	0.957	0.965	0.965	0.958	0.957	0.960	0.965	**0.967**	**0.967**
tae	0.460	0.460	0.400	0.420	0.360	0.400	0.400	0.300	**0.480**	0.460
data-banknote-authentication	0.967	0.947	0.976	0.974	0.976	0.947	0.991	0.982	0.985	**0.996**
fertility	0.879	0.848	0.909	0.879	0.788	0.879	0.879	0.909	**0.970**	**0.970**
seeds	0.886	0.871	0.857	0.857	0.871	0.886	0.886	0.886	0.857	**0.900**
balance-scale	0.768	0.754	0.787	0.802	0.763	0.768	0.744	0.773	0.430	**0.865**
segment	0.913	0.931	0.928	0.932	**0.942**	0.934	0.938	0.937	0.906	0.906
transfusion	0.733	0.709	0.737	0.745	0.733	0.717	0.721	0.700	0.672	**0.781**
phishing	0.857	**0.868**	0.852	0.848	0.819	0.857	0.859	**0.868**	0.848	0.848
yeast	0.504	0.496	0.490	0.512	0.512	0.473	0.527	0.512	**0.561**	0.559
wifi-localization	0.955	0.952	0.962	0.961	0.959	0.952	0.964	0.961	0.252	**0.976**
wall-following	0.998	0.997	0.996	0.996	0.994	0.997	0.997	0.999	0.997	**1.000**
vowel-context	0.615	0.535	0.670	0.679	0.443	0.612	0.639	**0.740**	0.694	0.618
spect	0.593	0.630	0.630	0.630	**0.667**	0.630	0.630	0.630	**0.667**	**0.667**
column	**0.796**	0.748	0.767	0.757	0.689	0.748	0.767	0.767	0.777	0.777
magic-04	0.795	0.788	**0.849**	**0.849**	0.799	0.787	0.812	0.837	0.839	0.842
wholesale-region	0.555	0.562	0.623	0.623	0.596	0.623	0.603	0.603	0.651	**0.705**
ecoli	0.718	0.727	0.800	0.800	0.773	0.727	0.745	0.764	0.864	**0.927**
cryotherapy	0.833	0.900	0.733	0.733	0.767	0.767	0.767	0.733	**0.900**	0.800
breast-tissue	0.543	0.514	0.543	0.543	0.600	0.543	0.514	0.571	**0.714**	**0.714**
wdbc	0.947	0.926	0.963	**0.968**	0.931	0.936	0.952	0.947	0.931	0.947
cardiotocography	0.969	0.954	0.963	0.962	0.950	0.953	0.963	0.964	0.774	**0.983**
mammographic-m	0.783	0.761	0.767	0.761	0.730	0.758	0.767	0.761	**0.830**	0.827
mfeat-factors-AttributeSelection	0.829	0.821	0.885	0.885	0.874	0.821	0.865	0.888	**0.927**	**0.927**
mfeat-pixel-AttributeSelection	0.830	0.845	0.876	0.888	0.914	0.855	0.883	0.892	**0.965**	**0.965**
mfeat-fourier-AttributeSelection	0.727	0.736	0.747	0.777	0.773	0.727	0.764	0.761	**0.827**	**0.827**
mfeat-morphological	0.668	0.671	0.694	0.686	0.661	0.658	0.665	**0.700**	0.403	0.689
mfeat-zernike	0.635	0.662	0.694	0.705	0.705	0.655	0.655	0.686	0.798	**0.824**
mfeat-pixel	0.836	0.823	0.865	0.892	0.898	0.845	0.877	0.885	**0.967**	**0.967**
optdigits	0.864	0.867	0.926	0.932	0.938	0.873	0.900	0.927	**0.981**	**0.981**
mfeat-karhunen	0.762	0.753	0.829	0.848	0.841	0.764	0.800	0.835	**0.958**	**0.958**
mfeat-fourier	0.692	0.673	0.724	0.745	0.742	0.674	0.712	0.730	**0.829**	**0.829**
Total: Best accuracy per Method	**1**	**3**	**1**	**2**	**2**	**1**	**2**	**3**	**28**	**35**

method was obtained by META in 15 databases (3 + 12), which is still far below the results obtained by our proposed methods.

When analysing the best performance delivered by both proposed methods, we can observe that they provided better performance with the real datasets (Table 4), while the existing ensemble methods delivered better performance with the artificial datasets (Table 5). It is a promising result for the proposed methods since the real datasets represent properly the real information of a classification problem, while the artificial dataset represent artificial manipulations of the original datasets.

In summary, we can observe that, in general, the performance of our proposed methods was superior to the existing methods (i.e. well-known in literature), showing that the use of meta-learning for the recommendation of the best ensemble structure can lead to robust classifier ensembles. Of the proposed methods, the sequential definition proposed in the ERM-3ML model seems to lead to more robust classifier ensembles than when using the ERM-ML model. We believe that this is due to the error propagation that occurs when we recommend all parameters of the ensemble structure (ERM-ML). In other words, the sequential recommendation is more appropriate to define the optimal ensemble structure.

TABLE 5. Results of method accuracy applied to artificial databases. List of competing methods: Multiple Classifier Behavior (MCB) [54], Overall Local Accuracy (OLA) [53], DES Performance (DESP) [56], K-Nearest Oracles Union (KNU) [55], Local class accuracy (LCA) [53], Classifier Rank (rank) [52], META-DES (META) [57], and our two methods: Ensemble Recommendation Method using Meta-Learning (ERMML) and Ensemble Recommendation Method using 3 steps Meta-Learning (ERM3ML).

	Methods									
Datasets (SMOTE)	**MCB**	**OLA**	**DESP**	**KNU**	**LCA**	**RANK**	**KNE**	**META**	**ERMML**	**ERM3ML**
iris0	0.940	0.960	0.960	0.960	0.960	0.960	0.960	0.960	**0.980**	**0.980**
iris1	**0.960**	**0.960**	0.940	0.940	**0.960**	**0.960**	**0.960**	0.940	0.940	0.940
iris2	0.920	**0.940**	**0.940**	**0.940**	0.900	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**
iris3	0.940	0.940	0.940	0.940	0.940	**0.960**	**0.960**	0.940	0.840	0.940
iris4	0.880	0.920	0.960	0.960	0.920	0.940	0.940	**0.980**	0.920	0.960
iris5	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**
iris6	**0.980**	**0.980**	0.960	0.960	**0.980**	**0.980**	**0.980**	**0.980**	**0.980**	**0.980**
iris7	**1.000**	**1.000**	**1.000**	**1.000**	0.860	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
iris8	0.880	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**	**0.900**
iris9	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	**0.940**	0.380	**0.940**
bupa1	0.675	0.728	0.781	0.763	0.728	0.702	0.728	0.781	0.781	**0.789**
bupa2	0.711	0.702	0.807	0.798	0.632	0.781	0.754	0.798	0.816	**0.868**
bupa3	0.772	0.789	0.754	0.798	0.737	0.807	0.816	0.798	0.798	**0.851**
bupa4	0.877	0.851	0.860	0.877	0.833	0.851	0.921	0.895	0.930	**0.947**
bupa5	0.895	0.868	0.851	0.904	0.816	0.904	0.930	**0.939**	0.912	0.912
bupa6	0.868	0.895	0.904	0.904	0.825	0.904	0.947	0.904	**0.965**	0.947
bupa7	0.912	0.904	0.930	0.930	0.895	0.939	0.939	0.904	0.842	**0.947**
bupa8	0.947	**0.974**	0.965	0.965	0.930	**0.974**	0.965	0.965	0.956	0.956
bupa9	0.982	0.974	0.956	0.956	0.982	0.974	0.965	0.956	**0.991**	0.982
bupaa	**0.982**	0.947	**0.982**	**0.982**	0.956	0.947	**0.982**	**0.982**	0.974	**0.982**
diabetes1	0.724	0.705	0.697	0.697	0.740	0.713	0.724	0.724	0.756	**0.776**
diabetes2	0.819	0.807	0.799	0.807	0.724	0.780	0.799	0.811	0.811	**0.839**
diabetes3	0.858	0.823	0.843	0.843	0.732	0.846	0.846	0.894	0.843	**0.906**
diabetes4	0.819	0.803	0.807	0.795	0.772	0.843	0.870	0.811	**0.882**	0.862
diabetes5	0.831	0.854	0.854	0.827	0.787	0.886	0.854	0.862	**0.929**	0.906
diabetes6	0.886	0.894	0.862	0.866	0.803	0.882	0.925	0.906	**0.933**	0.913
diabetes7	0.933	0.933	0.929	0.929	0.811	0.937	0.953	0.953	0.949	**0.957**
diabetes8	0.933	0.945	0.921	0.917	0.898	0.953	**0.957**	0.949	0.949	0.953
diabetes9	0.941	0.925	0.937	0.937	0.890	0.929	0.933	0.941	0.945	**0.976**
diabetesa	0.933	0.917	0.925	0.921	0.878	0.917	0.933	0.933	0.815	**0.941**
ecoli1	0.745	0.773	0.764	0.736	0.755	0.791	0.764	0.791	**0.864**	0.845
ecoli2	0.818	0.845	0.827	0.827	0.809	0.845	0.845	0.800	**0.855**	**0.855**
ecoli3	0.909	0.855	0.873	0.882	0.873	0.873	0.909	**0.918**	0.909	0.891
ecoli4	0.873	0.836	0.873	0.891	0.764	0.882	0.882	0.891	0.864	**0.909**
ecoli5	0.891	0.918	0.918	0.918	0.909	0.918	0.927	**0.955**	0.918	0.909
ecoli6	0.873	0.845	0.864	0.864	0.873	0.836	0.900	0.864	**0.973**	0.891
ecoli7	0.873	0.882	0.900	0.900	0.909	0.873	0.918	0.945	**0.955**	0.955
ecoli8	0.909	0.900	0.936	0.936	0.818	0.900	0.918	0.918	**0.964**	0.945
ecoli9	0.945	0.900	0.945	0.927	0.900	0.900	0.955	**0.973**	0.955	**0.973**
ecolia	0.927	0.927	0.918	0.918	0.918	0.945	0.945	0.945	0.945	**0.964**
glass1	0.648	0.648	0.732	0.746	0.577	0.634	0.662	0.704	0.690	**0.775**
glass2	0.676	0.634	0.718	0.732	0.479	0.620	0.634	0.676	0.648	**0.746**
glass3	0.761	0.690	0.648	0.648	0.606	0.690	0.690	0.732	**0.775**	0.761
glass4	0.606	0.620	0.676	0.676	0.606	0.676	**0.704**	0.676	0.690	0.690
glass5	0.718	0.775	0.620	0.761	0.718	**0.831**	0.803	0.775	**0.831**	0.746
glass6	0.761	0.775	0.563	0.634	0.606	0.775	0.775	0.620	0.775	**0.859**
glass7	0.634	0.563	0.634	0.620	0.423	0.620	0.662	0.620	0.746	**0.803**
glass8	0.732	0.718	0.648	0.648	0.648	0.718	0.718	0.732	0.761	**0.803**
glass9	**0.831**	0.676	0.662	0.662	0.704	0.704	0.775	0.761	0.775	0.817
glassa	0.746	**0.803**	0.676	0.746	0.620	0.761	0.746	0.761	**0.803**	0.789
Total: Best accuracy per Method	**7**	**9**	**6**	**6**	**5**	**10**	**11**	**12**	**19**	**29**

B. ANALYSIS OF ALL INVESTIGATED MODELS

Tables 4 and 5 present the accuracy results for each dataset individually. Based solely on these tables, it is not possible to observe the general performance of the analysed methods. Therefore, Figure 7 illustrates the boxplot of the accuracy results obtained by the all analyzed methods. Based on this boxplot, which was built from the data presented in Tables 4 and 5, it can be observed that our two proposed methods (the two rightmost boxes) present the best performance among all methods evaluated.

The proposed methods present the highest median value, having the lower quartile interval, superior to the others. On top of that, the inter-quartile intervals are larger than others, meaning a better performance of the proposed methods. It is also important to highlight that there is a slightly higher presence of outliers in the ERM-ML model, which may characterize a certain instability of this model in relation to others. The results obtained in Figure 7 only corroborates with the results shown in Tables 4 and 5.

As mentioned previously, the Friedman statistical test and the Nemenyi post-hoc test are also applied in order to analyze the obtained results from a statistical point of view. The Friedman test was applied to the performance of all ten methods and resulted in: $Friedman\ test = 245.08$,

TABLE 6. Summary of best accuracy results per model.

	Methods									
	MCB	OLA	DESP	KNU	LCA	RANK	KNE	META	ERMML	ERM3ML
Best accuracy	8	12	7	8	7	11	13	15	47	64

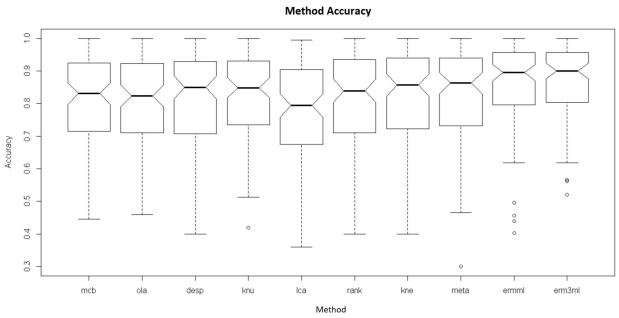

FIGURE 7. Boxplot of the accuracy results.

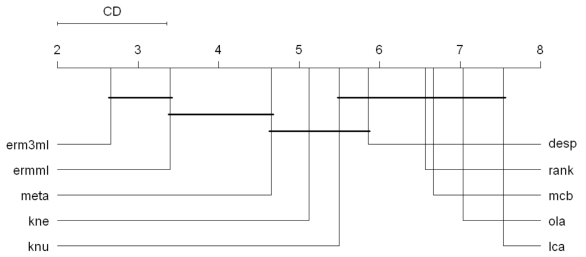

FIGURE 8. CD diagram of post-hoc test.

$df = 9$, $p - value < 2.2 * 10^{-16}$. It is important to emphasize that the Friedman test is applied directly to the accuracy values of all analyzed methods. In analyzing the Friedman test, we observed that the performance of all methods was statistically significant. This difference was detected by Friedman test since $p - value < 0.01$. The post-hoc test was then applied. Figure 8 presents the post-hoc test results through the critical difference (CD) diagram.

As it can be observed in Figure 8, both proposed methods outperformed all eight existing methods, being first and second in the ranking. The leftmost method, ERM-3ML, was statistically better than all existing ensemble methods. However, the CD diagram shows that there was no statistical difference between our two proposed methods. In relation to the ERM-ML method, it was statistically superior to seven existing ensemble methods. However, there was no statistical difference between this method and META. Additionally, the best ranked existing method was META, which outperformed all seven remaining methods. However, META was only statically superior to RANK, MCB, OLA and LCA.

In summary, we can state that ERM-3ML method was statistically better than all existing methods, and ERM-ML method was similar to META and statistically better than all seven methods. The results obtained in Figure 8 only corroborates with the idea that the use of meta-learning as a recommendation tool to define the best ensemble structure.

C. ATTRIBUTE DEPENDENCY ANALYSIS

Once we analyzed the performance of the ensemble systems in all 100 datasets, we also carried out an analysis to evaluate the dependence which might exist between each attribute and the performance of each analyzed method. The main motivation for performing this correlation analysis is in the fact that Meta-base has background information and there may be some information about a database that are more relevant in a meta-learner's decision making process than others.

This analysis aims to evaluate whether all attributes have similar influence or if some of them are more influential than others in ensembles' accuracy. In order to do this, Pearson correlation [60] has been used, and it is noteworthy that Pearson correlation coefficient measures the degree of linear correlation between two quantitative variables. It is a dimensionless index with values between -1 and 1, inclusive, which reflects the intensity of a linear relationship between two datasets. This coefficient, usually represented by a letter r, assumes only values between -1 and 1, where $r = 1$ means a perfect positive correlation between the two variables, $r = -1$ means a perfect negative correlation between the two variables, and $r = 0$ means that the two variables do not depend linearly on each other. However, there may be another dependency that is nonlinear, requiring data to be investigated by other means, this is better explained in [61].

The correlation between meta-attributes and the accuracy results of the obtained ensembles are presented in Table 7. Table 7 shows the main meta-attributes in the first column, and the analyzed methods are presented in the remaining columns. In addition, the highest correlation value for each meta-attribute is highlighted in bold.

From this table, it can be observed that the correlation between the majority of meta-attributes and the ensemble accuracy is weak (values close to 0). However, there is a high correlation between some meta-attributes and the ensemble accuracies. We highlight the the most correlated meta-attributes as follows: *Nr_attributes*: Number of attributes; *Nr_num_attributes*: Number of numerical attributes; and *SDRatio*: An M-Statistic transformation that evaluates information into the covariance structure of classes.

Among these attributes, the highest correlation was detected to the Nr_attributes meta-attribute. In order to analyze if this correlation is really apparent, a further detailed analysis must be done. To do this, we divided all datasets into three groups, based on the *Nr_attributes* meta-attribute, as lower (up to 5 attributes) central (between 6 and 47 attributes) and upper (equal or higher than 48 attributes).

TABLE 7. Correlation between meta-attributes and methods used for comparison.

Meta atributo	MCB	OLA	DESP	KNU	LCA	RANK	KNE	META	ERMML	ERM3ML
Nr_attributes	-0.104	-0.100	-0.025	-0.017	0.055	-0.107	-0.063	-0.030	**0.191**	**0.160**
Nr_sym_attributes	0.014	0.015	0.040	0.059	0.101	0.028	0.044	0.047	**0.109**	**0.101**
Nr_num_attributes	-0.187	-0.182	-0.094	-0.108	-0.050	-0.211	-0.162	-0.112	**0.159**	**0.120**
Nr_examples	0.055	0.052	0.135	**0.135**	**0.151**	0.050	0.073	0.106	0.110	0.125
Nr_classes	-0.169	-0.194	-0.127	-0.096	**-0.059**	-0.187	-0.135	-0.094	**-0.004**	-0.065
DefaultAccuracy	0.084	**0.088**	0.053	0.021	-0.043	**0.102**	0.069	0.030	0.039	-0.001
MissingValues_Total	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Lines_with_MissingValues_Total	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
Mean_Absolute_Skew	0.147	0.138	0.161	0.146	0.159	0.143	0.161	**0.165**	**0.193**	0.154
MeanKurtosis	0.147	0.138	0.161	0.146	0.159	0.143	0.161	**0.165**	**0.193**	0.154
NumAttrsWithOutliers	-0.294	-0.267	-0.288	-0.326	-0.215	-0.310	-0.301	-0.287	**0.080**	**0.056**
MStatistic	-0.041	-0.038	0.028	0.019	0.058	-0.047	-0.025	0.008	**0.121**	**0.114**
MStatDF	-0.016	-0.015	0.047	0.037	0.083	-0.029	0.001	0.030	**0.120**	**0.108**
MStatChiSq	0.169	0.190	**0.197**	0.159	**0.199**	0.181	0.125	0.122	0.013	0.142
SDRatio	0.129	0.152	0.158	0.123	**0.215**	0.141	0.087	0.110	0.073	**0.179**
Fract	**-0.058**	**-0.062**	-0.107	-0.114	-0.110	-0.070	-0.113	-0.127	-0.192	-0.132
Cancor	**-0.051**	**-0.064**	-0.098	-0.091	-0.069	-0.083	-0.108	-0.101	-0.202	-0.109
WilksLambda	**-0.129**	-0.136	-0.147	-0.165	-0.172	**-0.129**	-0.166	-0.184	-0.149	-0.163
BartlettStatistic	0.011	0.007	0.078	0.084	**0.105**	0.009	0.036	0.063	0.093	0.107
ClassEntropy	-0.133	-0.155	-0.086	-0.054	**0.002**	-0.152	-0.103	-0.056	**-0.014**	-0.030
EntropyAttributes	0.022	0.026	0.014	0.025	**0.045**	0.033	0.031	0.004	**0.048**	0.006
MutualInformation	0.003	0.012	-0.010	0.003	**0.036**	0.010	0.011	-0.023	**0.043**	-0.012
JointEntropy	-0.015	-0.007	-0.021	-0.006	**0.027**	-0.002	-0.002	-0.033	**0.039**	-0.016
Equivalent_nr_of_attrs	-0.288	-0.267	-0.311	-0.302	**-0.249**	-0.274	-0.291	-0.351	**-0.221**	-0.353
NoiseSignalRatio	-0.192	-0.187	-0.189	-0.185	-0.179	**-0.154**	-0.179	-0.203	**-0.159**	-0.228

TABLE 8. Best accuracy results by method.

Percentage of best accuracy results (attribute: Nr_attributes)										
Range	**MCB**	**OLA**	**DESP**	**KNU**	**LCA**	**RANK**	**KNE**	**META**	**ERMML**	**ERM3ML**
Lower (<=5 NR_attributes)	10%	20%	10%	10%	10%	20%	20%	10%	40%	90%
Central	9%	13%	8%	9%	8%	10%	13%	18%	41%	56%
Upper (>=48 NR_attributes)	0%	0%	0%	0%	0%	0%	10%	0%	100%	100%

In using this group division, the lower group is composed of 10 datasets, the upper group is composed of 10 databases and the central group is composed of 80 databases.

For each attribute size group, we calculate the proportion of winning for each analyzed method. The summary of this analysis is summarized in Table 8. Note that one proposed method, the ERM-ML method, had a proportion of the best results equals to 40 % in the lower group, 41 % in the central group and 100 % in the upper group. In addition, the ERM-3ML delivered a proportion of the best results equals to 90 % in the lower group, 56 % in the central group and 100 % in the upper group.

Thus, in all three attribute size groups, there is a predominance of the proposed methods in terms of good performance. However, the predominance of the proposed methods is stronger for the lower and upper groups. From this observation, we can state that the number of attributes is an important aspect that has a strong effect in the performance of all analyzed methods. However, this correlation is clear for the ERM-ML method, in which the proportion of winning increases as the number of attributes increases.

VII. CONCLUSION

As an attempt to solve the problem of defining the optimal ensemble structure, this paper proposed the use of meta-learning as a recommendation tool for different ensemble parameters, such as: pool size, classifier types and aggregation function. The main aim is to propose a recommendation system to provide accurate classifier ensembles. In the proposed approaches, the recommendation task is divided into two phases, training and evaluation. During the training phase, we extracted the characterization of a dataset, evaluated the best ensemble topology for this dataset and stored this information in a meta-database. In the evaluation phase, we applied a classifier to model the Meta-base dataset (meta-leaner). Then, we recommended the best pool size, classifier, and aggregation function for an unseen instance (classification dataset).

The proposed approach can be used to recommend the optimal ensemble structure and it can be used to any classification problem. Nonetheless, it can be applied only to classifier ensembles. In addition, its main drawback is the creation of the meta-base, that can be time consuming, but it is a limitation of a meta-learning recommendation system.

In order to assess the feasibility of the proposed approaches, an empirical analysis was conducted. In this experimental analysis, the performance of the proposed approaches were compared to eight well-known ensemble methods, applied to 100 well-known classification problems (datasets). The obtained results indicate that the proposed methods can indeed be used as a recommendation tool of an ensemble topology, providing the most accurate classifier ensembles for the majority of datasets. This results are promising, showing that the use of meta-learning to

recommend the ensemble structure is a robust way to achieve accurate classifier ensembles.

REFERENCES

[1] M.-W. Huang, C.-W. Chen, W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "SVM and SVM ensembles in breast cancer prediction," *Plos One J.*, vol. 12, no. 1, pp. 480–501, 2017.

[2] L. F. S. Coletta, E. R. Hruschka, A. Acharya, and J. Ghosh, "Using metaheuristics to optimize the combination of classifier and cluster ensembles," *Integr. Comput.-Aided Eng.*, vol. 22, no. 3, pp. 229–242, Jun. 2015.

[3] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Hoboken, NJ, USA: Wiley, 2004.

[4] S. Bashbaghi, E. Granger, R. Sabourin, and G.-A. Bilodeau, "Dynamic selection of exemplar-SVMs for watch-list screening through domain adaptation," in *Proc. 6th Int. Conf. Pattern Recognit. Appl. Methods*, 2017, pp. 738–745.

[5] P. R. L. de Almeida, E. J. da Silva Junior, T. M. Celinski, A. de Souza Britto, L. E. S. de Oliveira, and A. L. Koerich, "Music genre classification using dynamic selection of ensemble of classifiers," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2012, pp. 2700–2705.

[6] H. Xiao, Z. Xiao, and Y. Wang, "Ensemble classification based on supervised clustering for credit scoring," *Appl. Soft Comput.*, vol. 43, pp. 73–86, Jun. 2016.

[7] C. Porcel, A. Tejeda-Lorente, M. Martínez, and E. Herrera-Viedma, "A hybrid recommender system for the selective dissemination of research resources in a technology transfer office," *Inf. Sci.*, vol. 184, no. 1, pp. 1–19, 2012.

[8] S. Amara and R. R. Subramanian, "Collaborating personalized recommender system and content-based recommender system using TextCorpus," in *Proc. 6th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS)*, Mar. 2020, pp. 105–109.

[9] C. Troussas, A. Krouska, and M. Virvou, "Multi-algorithmic techniques and a hybrid model for increasing the efficiency of recommender systems," in *Proc. IEEE 30th Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2018, pp. 184–188.

[10] D. Di Nucci, F. Palomba, R. Oliveto, and A. De Lucia, "Dynamic selection of classifiers in bug prediction: An adaptive method," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 3, pp. 202–212, Jun. 2017.

[11] G. Giacinto, R. Perdisci, M. D. Rio, and F. Roli, "Intrusion detection in computer networks by a modular ensemble of one-class classifiers," *Inf. Fusion*, vol. 9, no. 1, pp. 69–82, Jan. 2008.

[12] A. M. P. Canuto, M. C. C. Abreu, L. D. M. Oliveira, J. C. Xavier, and A. M. Santos, "Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles," *Pattern Recognit. Lett.*, vol. 28, no. 4, pp. 472–486, 2007.

[13] C. Soares, P. B. Brazdil, and P. Kuba, "A meta-learning method to select the kernel width in support vector regression," *Mach. Learn.*, vol. 54, no. 3, pp. 195–209, Mar. 2004.

[14] C. Giraud-Carrier, R. Vilalta, and P. Brazdil, "Introduction to the special issue on meta-learning," *Mach. Learn.*, vol. 54, no. 3, pp. 187–193, Mar. 2004.

[15] B. F. Souza, "Meta-aprendizagem aplicada a classificacao de dados de expressao genica," M.S. thesis, Inst. Math. Comput. Sci., USP Sao Carlos, São Carlos, Brazil, 2010.

[16] A. Kalousis, "Algorithm selection via meta-learning," Ph.D. dissertation, Dept. Inform., Univ. Geneva, Geneva, Switzerland, 2002.

[17] R. R. Parente, A. M. P. Canuto, and J. C. Xavier, "Characterization measures of ensemble systems using a meta-learning approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Aug. 2013, pp. 1–8.

[18] D. S. C. Nascimento, "Configuração heterogenea de ensembles de classificadores: Investigaçãao em bagging, boosting e multiboosting," M.S. thesis, Stricto Sensu Postgraduate Division, Universidade de Fortaleza.UNIFOR, Fortaleza, Brazil, 2009.

[19] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, 2003.

[20] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.

[21] C. M. M. O. P. Soares, "Learning rankings of learning algorithms," M.S. thesis, Dept. Econ., Univ. Porto, Porto, Portugal, 2004.

[22] D. Michie, D. J. Spiegelhalter, C. C. Taylor, and J. Campbell, Eds., *Machine Learning, Neural and Statistical Classification*. Upper Saddle River, NJ, USA: Ellis Horwood, 1994.

[23] C. Troussas, A. Krouska, C. Sgouropoulou, and I. Voyiatzis, "Ensemble learning using fuzzy weights to improve learning style identification for adapted instructional routines," *Entropy*, vol. 22, no. 7, p. 735, Jul. 2020. [Online]. Available: https://www.mdpi.com/1099-4300/22/7/735

[24] Z. Zhu, Z. Wang, D. Li, Y. Zhu, and W. Du, "Geometric structural ensemble learning for imbalanced problems," *IEEE Trans. Cybern.*, vol. 50, no. 4, pp. 1617–1629, Apr. 2020.

[25] A. A. Feitosa Neto and A. M. P. Canuto, "An exploratory study of mono and multi-objective metaheuristics to ensemble of classifiers," *Int. J. Speech Technol.*, vol. 48, no. 2, pp. 416–431, Feb. 2018.

[26] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Proc. Adv. Neural Inf. Process. Syst.*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2015, pp. 2962–2970.

[27] P. Kordík, J. Černý, and T. Frýda, "Discovering predictive ensembles for transfer learning and meta-learning," *Mach. Learn.*, vol. 107, no. 1, pp. 177–207, Jan. 2018.

[28] L. Kotthoff, C. Thornton, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 826–830, Jan. 2017.

[29] W. A. Albukhanajer, Y. Jin, and J. A. Briffa, "Classifier ensembles for image identification using multi-objective Pareto features," *Neurocomputing*, vol. 238, pp. 316–327, May 2017.

[30] M. Wistuba, N. Schilling, and L. Schmidt-Thieme, "Automatic Frankensteining: Creating complex ensembles autonomously," in *Proc. SIAM Int. Conf. Data Mining*, Houston, TX, USA, Apr. 2017, pp. 741–749.

[31] L. Zhang, W. Srisukkham, S. C. Neoh, L. C. Peng, and D. Pandit, "Classifier ensemble reduction using a modified firefly algorithm: An empirical evaluation," *Expert Syst. Appl.*, vol. 93, pp. 395–422, Mar. 2018.

[32] D. S. C. Nascimento, A. M. P. Canuto, and A. L. V. Coelho, "An empirical analysis of meta-learning for the automatic choice of architecture and components in ensemble systems," in *Proc. Brazilian Conf. Intell. Syst. (BRACIS)*, Oct. 2014, pp. 1–6.

[33] J. R. Rice, "The algorithm selection problem," in *Proc. Memorial Lecture Comput. Sci. Conf.* in Advances in Computers, vol. 15, M. Rubinoff and M. C. Yovits, Eds. Amsterdam, The Netherlands: Elsevier, 1976, pp. 65–118.

[34] L. Rendell and H. Cho, "Empirical learning as a function of concept character," *Mach. Learn.*, vol. 5, no. 3, pp. 267–298, Aug. 1990.

[35] R. D. King, C. Feng, and A. Sutherland, "STALOG: Comparison of classification algorithms on large real-world problems," *Appl. Artif. Intell.*, vol. 9, no. 3, pp. 289–333, May 1995.

[36] P. B. Brazdil, C. Soares, and J. P. da Costa, "Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results," *Mach. Learn.*, vol. 50, no. 3, pp. 251–277, Mar. 2003.

[37] A. Kalousis, J. Gama, and M. Hilario, "On data and algorithms: Understanding inductive performance," *Mach. Learn.*, vol. 54, no. 3, pp. 275–312, Mar. 2004.

[38] E. Bernado-Mansilla and T. K. Ho, "Domain of competence of XCS classifier system in complexity measurement space," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 82–104, Feb. 2005.

[39] S. Ali and K. A. Smith, "On learning algorithm selection for classification," *Appl. Soft Comput.*, vol. 6, no. 2, pp. 119–138, Jan. 2006.

[40] K. A. Smith, F. Woo, V. Ciesielski, and R. Ibrahim, "Modelling the relationship between problem characteristics and data mining algorithm performance using neural networks," in *Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, and Complex Systems*, C. Dagli *et al.*, Eds. 2001, pp. 357–362.

[41] K. A. Smith, F. Woo, V. Ciesielski, and R. Ibrahim, "Matching data mining algorithm suitability to data characteristics using a self-organizing map," in *Hybrid Information Systems*, A. Abraham and M. Köppen, Eds. Berlin, Germany: Physica-Verlag HD, 2002, pp. 169–179.

[42] A. R. S. Parmezan, H. D. Lee, and F. C. Wu, "Estudo preliminar da construção de um modelo de recomendação de algoritmos de seleção de atributos utilizando meta-aprendizado," in *Proc. 20th Simpósio Internacional de Iniciação Científica da Universidade de São Paulo (SIICUSP)*, 2012, p. 1.

[43] G. Wang, Q. Song, H. Sun, X. Zhang, B. Xu, and Y. Zhou, "A feature subset selection algorithm automatic recommendation method," *J. Artif. Intell. Res.*, vol. 47, pp. 1–34, May 2013.

[44] S. Shilbayeh and S. Vadera, "Feature selection in meta learning framework," in *Proc. Sci. Inf. Conf.*, Aug. 2014, pp. 269–275.

[45] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel, "Automatic classifier selection for non-experts," *Pattern Anal. Appl.*, vol. 17, no. 1, pp. 83–96, Feb. 2014.
[46] A. Filchenkov and A. Pendryak, "Datasets meta-feature description for recommending feature selection algorithm," in *Proc. Artif. Intell. Natural Lang. Inf. Extraction, Social Media Web Search FRUCT Conf. (AINL-ISMW FRUCT)*, Nov. 2015, pp. 11–18.
[47] A. Zabashta, I. Smetannikov, and A. Filchenkov, "Rank aggregation algorithm selection meets feature selection," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner, Ed. Cham, Switzerland: Springer, 2016, pp. 740–755, doi: 10.1007/978-3-319-41920-6_56.
[48] R. A. D. Silva, A. M. D. P. Canuto, J. C. Xavier, Jr., and T. B. Ludermir, "Using meta-learning in the selection of the combination method of a classifier ensemble," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.
[49] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, CA, USA, 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[50] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, and S. García, "KEEL data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework," *J. Multiple-Valued Logic Soft Comput.*, vol. 17, nos. 2–3, pp. 255–287, 2011.
[51] R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti, "Dynamic classifier selection: Recent advances and perspectives," *Inf. Fusion* vol. 41, pp. 195–216, Sep. 2018.
[52] M. Sabourin, A. Mitiche, D. Thomas, and G. Nagy, "Classifier combination for hand-printed digit recognition," in *Proc. 2nd Int. Conf. Document Anal. Recognit. (ICDAR)*, Oct. 1993, pp. 163–166.
[53] K. Woods, K. Bowyer, and W. P. Kegelmeyer, "Combination of multiple classifiers using local accuracy estimates," in *Proc. CVPR IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 1996, pp. 391–401.
[54] G. Giacinto and F. Roli, "Dynamic classifier selection based on multiple classifier behaviour," *Pattern Recognit.*, vol. 34, no. 9, pp. 1879–1881, Sep. 2001.
[55] A. H. R. Ko, R. Sabourin, and A. S. Britto, Jr., "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognit.*, vol. 41, no. 5, pp. 1718–1731, May 2008.
[56] T. Woloszynski, M. Kurzynski, P. Podsiadlo, and G. W. Stachowiak, "Hang," *Inf. Fusion*, vol. 13, no. 3, pp. 207–213, 2012.
[57] R. M. O. Cruz, R. Sabourin, G. D. C. Cavalcanti, and T. Ing Ren, "META-DES: A dynamic ensemble selection framework using meta-learning," *Pattern Recognit.*, vol. 48, no. 5, pp. 1925–1935, May 2015.
[58] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[59] R. Vilalta, C. G. Carrier, and P. Brazdil, *Meta-Learning*. Boston, MA, USA: Springer, 2005, ch. 33, pp. 731–748.
[60] P. Y. Chen and P. M. Popovich, "Correlation: Parametric and nonparametric measures," in *Sage University Papers Series*. Newbury Park, CA, USA: Sage, 2002, pp. 137–139.
[61] P. Schober, C. Boer, and L. A. Schwarte, "Correlation coefficients: Appropriate use and interpretation," *Anesthesia Analgesia*, vol. 126, no. 5, pp. 1763–1768, May 2018.

ROBERCY ALVES DA SILVA received the Ph.D. degree from the Federal University of Rio Grande do Norte, in 2020. He is currently a Senior Lecturer in electronics with the Federal Institute of Rio Grande do Norte. His research interests include classifier combination, meta-learning, the Internet of Things, computer networks, digital systems, electronic systems, industrial and residential automation, and fault detection.

ANNE MAGÁLY DE PAULA CANUTO received the Ph.D. degree from the University of Kent, in 2001. She is currently an Associate Professor with the Department of Informatics and Applied Mathematics, Federal University of Rio Grande do Norte, Brazil. She has published several articles in scientific journals and conferences. Her interests include pattern recognition, clustering algorithms, biometrics, classifier combination, semi-supervised learning, and multi-agent systems.

CEPHAS ALVES DA SILVEIRA BARRETO received the master's degree in software engineering from the Federal University of Rio Grande do Norte, Natal, Brazil, in 2018, where he is currently pursuing the Ph.D. degree in systems and computation. His research interests include machine learning, automotive data, and intelligent applications.

JOÃO CARLOS XAVIER-JUNIOR received the master's degree from the University of Kent, U.K., in 2001, and the Ph.D. degree from the Federal University of Rio Grande do Norte, Brazil, in 2012. He is currently an Associate Professor with the Digital Metropolis Institute, Federal University of Rio Grande do Norte. He has published several articles in scientific journals and conferences. His research interests include semi-supervised learning methods, classifier ensemble methods, clustering and distance metrics, automated machine learning, and evolutionary algorithms optimization.