International Journal of Computational Intelligence Systems (2022) 15:74
https://doi.org/10.1007/s44196-022-00134-0

RESEARCH ARTICLE

Continuous Dynamic Update of Fuzzy Random Forests

Jordi Pascual-Fontanilles[1] · Aida Valls[1] · Antonio Moreno[1] · Pedro Romero-Aroca[2]

Received: 4 March 2022 / Accepted: 18 August 2022

Abstract
Fuzzy random forests are well-known machine learning classification mechanisms based on a collection of fuzzy decision trees. An advantage of using fuzzy rules is the possibility to manage uncertainty and to work with linguistic scales. Fuzzy random forests achieve a good classification performance in many problems, but their quality decreases when they face a classification problem with imbalanced data between classes. In some applications, e.g., in medical diagnosis, the classifier is used continuously to classify new instances. In that case, it is possible to collect new examples during the use of the classifier, which can later be taken into account to improve the set of fuzzy rules. In this work, we propose a new iterative method to update the set of trees in the fuzzy random forest by considering trees generated from small sets of new examples. Experiments have been done with a dataset of diabetic patients to predict the risk of developing diabetic retinopathy, and with a dataset about occupancy of an office room. With the proposed method, it has been possible to improve the results obtained when using only standard fuzzy random forests.

Keywords Fuzzy sets · Random forest · Dynamic learning models · Classification methods

Abbreviations

FRF	Fuzzy random forest
DR	Diabetic retinopathy
FDT	Fuzzy decision tree
OC	Occupancy

✉ Jordi Pascual-Fontanilles
jordi.pascual@urv.cat

Aida Valls
aida.valls@urv.cat

Antonio Moreno
antonio.moreno@urv.cat

Pedro Romero-Aroca
pedro.romero@urv.cat

1 ITAKA-Intelligent Technologies for Advanced Knowledge Acquisition, Departament d'Enginyeria Informàtica i Matemàtiques, Universitat Rovira i Virgili, Avda. Paisos Catalans, 26, 43007 Tarragona, Catalunya, Spain

2 Servei d'Oftalmologia, Hospital Universitari Sant Joan de Reus, Institut d'Investigació Sanitària Pere Virgili (IISPV), Universitat Rovira i Virgili, Tarragona, Catalunya, Spain

1 Introduction

Building accurate decision support systems is hard. In the last decade, it has become common to use classifiers based on Machine Learning for this task. Usually these systems face a trade-off between sensitivity and specificity. In imbalanced datasets, it is difficult to simultaneously have a high sensitivity and specificity, as one class is over-represented. In this setting, classifiers have trouble identifying the objects of the minority class. This is a common problem in medical diagnosis systems, where usually the set of people suffering a disease is small in comparison with the healthy people. However, in this case, it is particularly interesting to have a low number of false negatives, that is, a good sensitivity. If the classifier tends to overestimate the negative (healthy) class, sensitivity decreases.

In this work, we consider a scenario in which an initial classifier has been built with a reasonably large dataset of historical data, but it is not accurate enough to distinguish properly the categories. After a certain period of time, as the institution (e.g., company, hospital) is able to gather some more labeled data, we want to revise the initial classifier to improve its performance by taking advantage of the new examples.

The kind of classifiers we have considered are Random Forests (RF), which are ensemble learning methods that

Published online: 06 September 2022

combine multiple Decision Trees (DT). They have shown a great performance in classification tasks in comparison with other techniques [1]. In domains with uncertainty or imprecision, we can use Fuzzy Random Forests (FRF). Their ensemble is constructed using Fuzzy Decision Trees (FDT), where the rules are defined on fuzzy linguistic variables and many rules can be activated to different levels depending on the inputs. Despite their good accuracy, these models still struggle in the presence of imbalanced datasets, as explained before.

In this paper, we propose a novel method to take advantage of the new data that may arrive to a decision support system based on a Fuzzy Random Forest. When a set of data big enough has been gathered, these new examples will be used to make an update on the FRF model, with the aim of improving its performance. The method may also reuse the training examples that the FRF is not able to classify correctly. Even though it has been tested on a FRF, the method is suited to be used with standard random forests.

The proposed updating model has been evaluated on two different datasets. The first one consists on the detection of the risk of developing Diabetic Retinopathy (DR). As a consequence of diabetes, the blood vessels of the eye may break and generate small blood spots, hemorrhages and exudates. These lesions produce vision loss and may even cause blindness if they are not detected and treated at an early stage. The risk of developing DR can be calculated from some clinical data of the patient, including blood analysis results. This classification problem is currently being solved using the Retiprogram system, which is based on a FRF. In some previous works other models were tested in this problem, obtaining worse results [2, 3]. The use of fuzzy input attributes allows the system to reason in a way closer to humans. The linguistic variables obtained from the fuzzified attributes are also much more interpretable by medical experts. Moreover, they avoid reasoning with precise numerical data when it is not required by the problem. To assess the risk of developing Diabetic Retinopathy, doctors reason qualitatively on the attribute values (e.g., age: child/young/old, body mass: underweight/normal/overweight, hypertension: good control/bad control, etc.). A difference of one year or of one kilogram makes no difference in the diagnosis, which is done at a more general scale of measurement (with labels). However, the input data are precise and numerical, so fuzzification is a proper procedure to move from the numerical scale to the linguistic scale of measurement.

This system is being tested by a group of ophthalmologists at Hospital Sant Joan de Reus. The general results are good (with a sensitivity and a specificity over 75%), but there are still many misclassifications. Errors are mainly due to the inherent ambiguity of the training examples (very similar patients can belong to different classes) and to the high imbalance between both classes (more than 90% of diabetic patients do not develop DR). Using the proposed updating method, the data from the new patients which are treated at the hospital has been used to update the base FRF model.

The second dataset used in this work is related to the detection of the occupancy of an office room [4]. A FRF classifier has also been used to determine if the office room is occupied or not. In this case, the reasoning with environmental variables (temperature, light, CO_2 concentration, etc.) can be done in a qualitative way on a fuzzy scale rather than a numerical one. The data are also highly imbalanced towards the negative class. In this case, the dataset has been split to simulate the arrival of new labeled data to the system from time to time.

Experimental results have been obtained on both datasets. In both cases, the numerical input variables have been fuzzified. The output of the FRF classifier is the class with a higher activation. This allows using standard metrics to assess the performance of the classifier. The weighted balanced accuracy has been used as the performance metric, allowing to face the trade-off between the sensitivity and the specificity caused by the class imbalance. Several tests have been performed to check the performance of the model during the multiple iterations of the updating method.

The rest of the paper is organized as follows. Section 2 presents other approaches used to update fuzzy random forests, consisting on adding weights to the decision trees, or on building trees dynamically from streaming data. In Sect. 3, we introduce the proposed method for iteratively updating the trees in a FRF. In Sect. 4, we present the datasets and we discuss the obtained experimental results. Finally, Sect. 5 presents the conclusions and the lines of future work.

2 Related Work

The optimization of classification models based on Random Forests has been studied in the literature. Even though most techniques are not fuzzy, they could also be applied to Fuzzy Decision Trees or Fuzzy Random Forests. We can distinguish two main approaches: adding weights to the FRF, or building online FRFs. The former is summarized in Sect. 2.1, whereas the latter is presented in Sect. 2.2. Finally, in 2.3, an analysis of these methods is performed.

2.1 Adding Weights to Fuzzy Random Forests

Weighting some of the components of the classification model is one of the ways to achieve a better performance. During the training stage, weights may be added in the model in four different ways. The first one consists of adding weights at the last step of the FRF classifier, when a voting procedure is made to find the majority class [5–7]. Each tree on the ensemble has a weight which corresponds to its accuracy. The accuracy is obtained by calculating the performance of the tree on the out-of-bag samples. Other possibilities consist on changing the weights during the training stage. For instance, Dogan and Birant [8] proposed initializing all the weights to the same value, and reward the best performing trees on the validation set formed by the out-of-bag samples. Zhukov et al. [9] added a pruning step to replace the worst decision tree, so the ensemble can handle concept drift. Decision trees may also have a sliding window of stored samples. Each time a new sample has to be evaluated, similar samples from the sliding window are used to recompute the weights based on their errors.

The second option consists on weighting the samples. Kim et al. [10] proposed weighting the samples according to the complexity of classifying them correctly. The trees also have weights, which are computed using the ones on the samples already classified. Yang and Yin [11] considered finding the weight of the decision trees as an optimization problem. For a certain number of epochs, both the weights on the samples and the decision trees are updated to optimize the model.

The third possibility, proposed by Zhong et al. [12], assigns weights to the leaves of the decision trees. That is, each of the rules of the ensemble has a weight based on some performance metric. For regression problems, this method obtains better results than just weighting the decision trees. A similar approach is proposed by Khan et al. [13], in which they applied weights to the rules on fuzzy decision trees. The rule weight is considered to be the certainty factor, which is computed for each branch of the FDT using the training data. The fuzzy rule with the maximum membership value for a sample to be classified is the one which decides the final class.

Finally, there are weighting methods that use different weights for each of the possible output classes. For instance, Zhu et al. [14], Livieris et al. [15] and Utkin et al. [16] use this approach to compensate imbalanced datasets.

2.2 Online Fuzzy Random Forests

The second kind of methods deal with the so-called online random forests. They differ from conventional RFs in that they are dynamically constructed and optimized using streams of continuous data. In this case, the training data arrives and is processed continuously. As the new training samples are not available from the beginning, the methods are not focused on improving an existing model, but on adapting the current one. Gomes et al. [17] reviewed several methods for data stream classification using ensemble-based methods, and proposed a taxonomy to classify them.

Incremental decision trees are one type of online learning methods. This type of decision trees can be grown in an online fashion, i.e., their rules can be updated using new data examples. For instance, Kalles and Morris [18] and Utgoff et al. [19] proposed variants of ID3 which are incremental. Regarding fuzzy approaches, Guetova et al. [20] proposed a fuzzy incremental variant of ID3. Ichihashi et al. [21] also proposed an incremental variant of ID3, Neuro-Fuzzy ID3, in which they considered the membership function as a three-layered neural network. Isazadeh et al. [22] and Pecori et al. [23] also proposed different approaches based on Very Fast Decision Trees to train an ensemble of fuzzy incremental decision trees, using streaming data.

Saffari et al. [24] combined online bagging techniques with extremely randomized forests to build an online random forest. The trees on this random forest grow when new data are fed into the model. A new branch on the tree is created when there are enough samples on a node, and they are good enough to classify new samples. Similar proposals of online random forests include Mondrian Forests [25] and Adaptive Random Forests [26]. They are also based on growing the trees' branches (i.e., the rules of the trees) with the arrival of new training samples. Some incremental approaches also drop some members of the ensemble, and create new ones. Their objective is to handle concept drifts, and their focus is on processing streams of data.

2.3 Analysis of the Related Work

The two analyzed approaches to improve the construction of random forests are very different. On the one hand, the weighting methods are applied during the training of the model. They do not need new data because they use the out-of-bag training samples. As the optimization is done during the training, the core of the model is not modified a posteriori. Published results show that these approaches are able to improve the performance of a standard random forest. Despite updating the weights using new data is rarely done, these weighting methods could be used to update the core model using new data. However, they would not learn new patterns as these techniques do not create new rules.

On the other hand, the main drawback of online random forests is that they need more data than standard random forests to achieve a similar performance. They are not designed to update and improve an existing model, but to construct it while it is used. They are well suited for applications which have to process continuous streams of data, as they need large amounts of data to be trained.

The main difference of these approaches with the method proposed in this paper is that we want to first create a classification system with a FRF, and later this FRF will be dynamically updated from time to time using new small sets of data. Some similarities exist with on-line methods. Specifically, with incremental approaches, according to the taxonomy proposed in [17]. They are designed to work on data streaming and are focused on handling concept drift. In contrast, the kind of problems we aim to solve do not include the use of data streams, and they do not have any concept drift. They are explained in more detail in Sect. 4.1.

3 Proposed Method

This paper presents a new method to update a Fuzzy Random Forest using a new set of incoming data. It consists on modifying the set of trees that compose the FRF model taking into account new examples that were not initially available. The goal is to try to increase the performance by updating the classification model without retraining it from scratch. This updating process will be performed after collecting a sufficiently big set of new cases that can be used as examples for improving the model. The proposed architecture is illustrated in Fig. 1.

As a first dynamic component, we have the new data collected for updating the trees in the Random Forest. To improve the results and to enlarge the collected data in each update iteration, the use of previous misclassified examples (i.e., errors) is proposed. Three different ways of dealing with errors are studied. They are called NoError, ErrorLT (Errors from the Last Test) and AllData &ErrorLT, depending on which error examples are used during the dynamic updating. NoError and ErrorLT use the new data that arrived at the system in the current update step. In contrast, AllData &ErrorLT uses all the new data that arrived at the system in all the previous updating steps. Regarding the misclassified examples, NoError does not include them, whereas ErrorLT and AllData &ErrorLT include the misclassified examples of the previous update step.

As a second dynamic component, we consider the ensemble voting procedure of the Fuzzy Random Forest. Two methods are usually applied. The first one is the majority voting, in which the most voted prediction among the Fuzzy Decision Trees in the ensemble is the final answer. The second one is the weighted voting, in which the vote

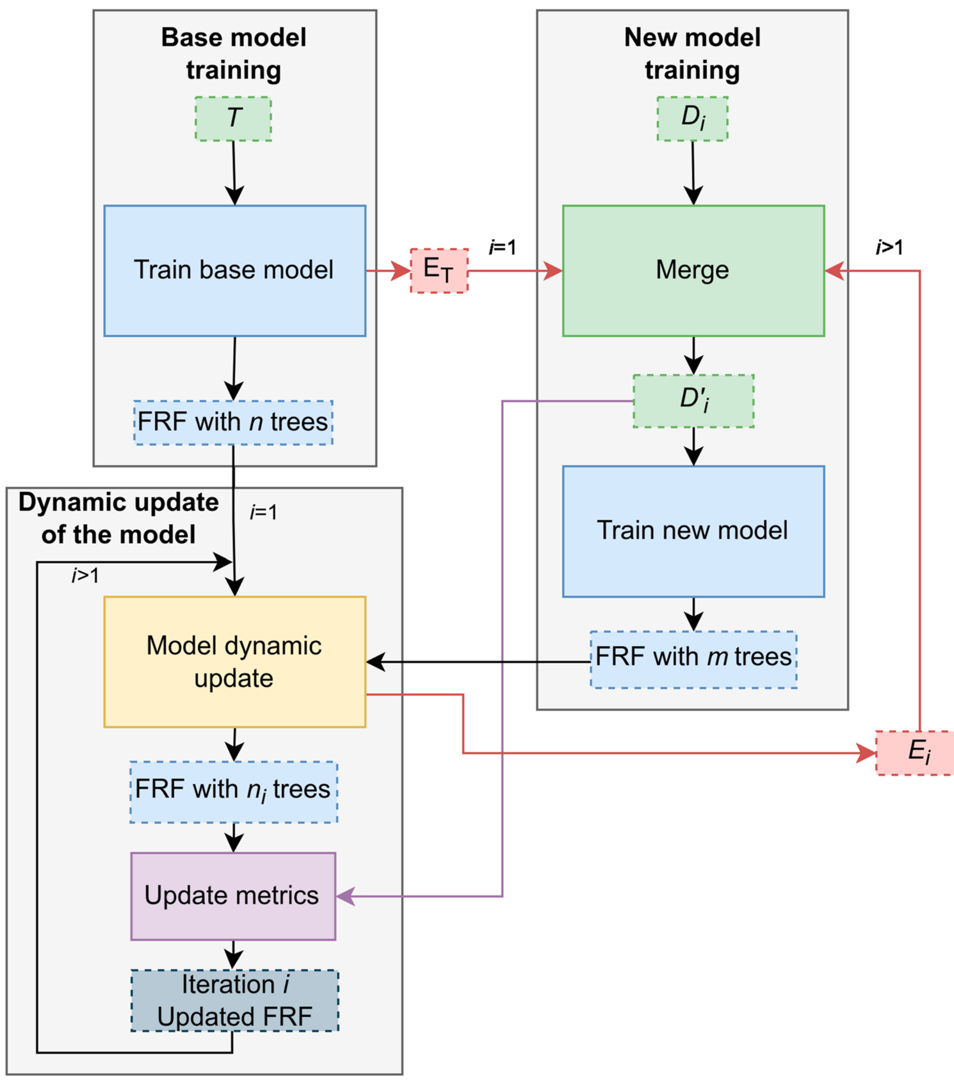

Fig. 1 Architecture of the iterative learning of Fuzzy Random Forests

of each Fuzzy Decision Tree is weighted according to its performance.

As a third dynamic component, we consider the possibility of updating the metrics of the Random Forest using the new collected data. The values of these metrics are used in the proposed method to update the model, and as weights in the weighted voting.

Algorithm 1 FRF iterative update algorithm

Input: T, $method$, $doUpdateWeights$

1: $updatedModel, E_T \leftarrow modelTrain(T)$ // Base model training
2: $i \leftarrow 1$ // Current update iteration
3: $D_i \leftarrow waitForNewData()$ // Data used to update the model
4: $E_{i-1} \leftarrow none$ // Error data with misclassified samples
5: **if** $method$ is $ErrorLT$ or $method$ is $AllData\&ErrorLT$ **then**
6:     $E_{i-1} \leftarrow E_T$
7: **end if**
8: **repeat** // Iterative model update process
9:     **if** $method$ is $NoError$ **then**
10:         $D'_i \leftarrow D_i$
11:     **else if** $method$ is $ErrorLT$ **then**
12:         $D'_i \leftarrow merge(D_i, E_{i-1})$
13:     **else if** $method$ is $AllData\&ErrorLT$ **then**
14:         $D'_i \leftarrow merge(D_{0..i}, E_{i-1})$
15:     **end if**
16:     $newModel \leftarrow modelTrain(D'_i)$
17:     $updatedModel, E_i \leftarrow modelUpdate(updatedModel, newModel)$
18:     **if** $doUpdateWeights$ is $true$ **then**
19:         $updatedModel \leftarrow updateWeights(updatedModel, D'_i)$
20:     **end if**
21:     $i \leftarrow i + 1$
22: **until** $(D_i \leftarrow waitForNewData()) == none$ // Repeat if there is new data

Algorithm 1 shows the procedure of construction and update of the FRF. The method is composed of three steps. The first one consists in the training of the base classification model, and it is only run once at the beginning. The other two steps are executed iteratively each time we collect a new set of examples and the model can be updated. These three steps are commented in more detail as follows:

1. *Base model training* The first step consists on training the base model with a large training dataset, T, which contains labeled examples (line 1). With a learning algorithm for Fuzzy Random Forests, we obtain n FDTs, where n is a large number, usually more than 100. During the construction process, the out-of-bag samples of each FDT are used to compute two metrics for each of them, the specificity and the sensitivity, which are stored on the FDT. Those metrics have two main purposes. The first one is to be used in the weighted voting if this option is selected. They are also used in the third step of the proposed method, in the update process. The obtained classification model should be validated with a testing dataset to ensure its good performance (this step is not shown in Fig. 1). After creating the base FRF, the training dataset can be used for testing, and the samples that are not correctly classified are stored in a file E_T. Those error samples E_T are used in the following step in the *ErrorLT* and *AllData &ErrorLT* versions.
2. *New model training* Every time enough new samples D_i have been gathered, usually around 200 samples, a new training iteration i is performed (lines 2–16). The merge process generates the dataset used to update the FRF. Its output, D'_i, depends on the method version used:

- The NoError version does not merge anything with D_i (line 10).
- In contrast, the *ErrorLT* version merges the errors data from previous iterations with D_i (line 12). For the first training iteration $i = 1$, the E_T errors samples are merged. In further iterations, the merged errors samples, E_i, are generated in the third step of the method.
- The *AllData &ErrorLT* version is an extension of *ErrorLT*. It additionally merges the training data from all previous iterations, $D_{0..i}$ (line 14).

Once the merge process is completed, the D'_i samples are used to train a new FRF (line 16). The difference between the base model and this newly generated one is the number of trees. Because the size of the new training set is small, we train a lower amount of FDTs m, with $m <<< n$, usually around 20 trees. Their out-of-bag samples are also used to compute the aforementioned metrics for each of these new trees. They are also used for the weighted voting, and in the third step of the proposed method.

3. *Dynamic update* The current model is updated in this step (lines 17–20). If the iteration is the first one ($i = 1$), the base model is updated. In further iterations where $i > 1$, the model being updated is the resulting model from the previous iteration $i - 1$. The m new FDTs trained in the previous step are used to update the current model (line 17). To do so, the m FDTs are added to the current model. To improve the performance of the updated FRF, the worst FDTs from those $n_{i-1} + m$ trees are removed. The number of trees being removed is fixed by a certain percentage p. The updated model will have a total of $(1 - p/100) \times (n_{i-1} + m)$ FDTs. To sort the trees and keep the best ones, a quality metric is used, the weighted balanced accuracy (1). It is defined as a weighted average between specificity and sensitivity, with a weighting factor α.

$$\mathrm{BA} = \alpha \cdot \mathrm{sensitivity} + (1 - \alpha) \cdot \mathrm{specificity}. \quad (1)$$

After pruning the worst trees, an additional update metrics process can optionally be performed (lines 18–20). When this option is selected, the metrics computed using the out-of-bag samples are updated. The training data D'_i of the current iteration i are used to compute the quality metrics for each of the FDTs. Instead of replacing the old metrics, the average between the old and the new metric is computed $\mathrm{NewMetric} = (\mathrm{CurrentMetric} + \mathrm{UpdatedMetric})/2$. This allows a more gradual update of the metrics.

The resulting FRF with n_i trees is set as the current model, and it is taken as the new model to be used until a new set of cases is available, and a new update iteration starts.

The errors of the updated FRF model on the D'_i dataset may also be retrieved and stored in E_i as it was done for the base model with E_T. In the case of using the *ErrorLT* or *AllData* &ErrorLT versions, in the next iteration, the new samples D_i are merged with those error cases E_i to enlarge the training dataset of the subsequent iteration.

Table 1 Diabetic retinopathy patients data

Dataset	Training	Validation	Testing	Total
DR = 0 samples	1376 (72%)	380 (63%)	863 (78%)	2619
DR = 1 samples	537 (28%)	222 (37%)	240 (22%)	999
Total samples	1913	602	1103	3618

Table 2 Office room occupancy data

Dataset	Training	Validation	Testing	Total
OC = 0 samples	3064 (78%)	1583 (79%)	1293 (79%)	5940
OC = 1 samples	844 (22%)	417 (21%)	336 (21%)	1597
Total samples	3908	2000	1629	7537

The use of the sets of wrongly classified examples E_i is optional. It is only used in the *ErrorLT* and *AllData &ErrorLT* versions of the iterative method. Their use has two purposes. On the one hand, to increase the size of the training set D'_i and, on the other hand, to show again these wrongly classified cases to the learning model in order to be able to build new rules that cover them appropriately. In that way, the model is learning from the past errors. In the next section, the effects of the diverse configurations of the proposed method are studied.

4 Experimental Results

In this section, the obtained experimental results are shown. In Sect. 4.1 the tested datasets are explained and analyzed. In Sect. 4.2 the selection of the method parameters is explained. Section 4.3 shows the obtained results and analyses them. Finally, in Sect. 4.4, an in-depth analysis of some results is performed, to study how the proposed method modifies the FRF model.

4.1 Datasets

Two different datasets have been used to test the proposed iterative method for updating a FRF. The first one is the diabetic retinopathy (DR) risk detection problem. This is a private dataset from Hospital Sant Joan de Reus, located in

Catalonia, Spain. It is a binary classification problem with two labels: DR = 1 means a high risk of suffering from diabetic retinopathy (i.e., positive class), whereas DR = 0 means a low risk (i.e., negative class). The experiments have been performed using real data from diabetic patients. This data includes 9 different attributes, 6 numerical (Age, Evolution time of diabetes, HbA1c, CKD-EPI, Microalbuminuria and Body Mass Index) and 3 categorical (Sex, Medical Treatment and Hypertension). The target attribute is the label of the class DR = 0 or DR = 1.

The second one is the occupancy (OC) dataset [4], in which the occupancy of an office room is predicted. The dataset is publicly available at the UCI Machine Learning Repository. It is also a binary classification problem with two labels. OC = 0 means the office room is not occupied (i.e., negative class), whereas OC = 1 means the office room is occupied (i.e., positive class). The occupancy dataset has 6 different attributes, 5 numerical (Temperature, Humidity, Light, CO_2 and Humidity Ratio) and 1 categorical (Date). The target attribute is the label of the class OC = 0 or OC = 1.

The data from both problems are split in three different datasets: training, validation and testing. The training dataset, T, is used to train the base FRF model. It is used to create the model with the largest number of trees (100 trees); hence, it is the dataset with more samples.

The validation set is used to simulate the new data that would arrive to the system from time to time. We split the validation set in chunks of 200 samples. Each of them is used in a different iteration i during the dynamic updating process. From each of these small new training datasets, D_i, the system generates 20 new trees. Then, the dynamic updating step is done, obtaining the FRF model M_i.

Finally, the testing set is used after each iteration to check the performance of the new FRF M_i. Note that the samples from this testing dataset are not included in the error sets; thus, the model is never trained using them.

Tables 1 and 2 show the splitting of the data among the three datasets for the diabetic retinopathy and occupancy problems, respectively. It can be seen that both datasets are highly imbalanced towards the negative class.

To obtain the experimental results, we used the aforementioned datasets to build a FRF for each of the problems. Because of the use of a FRF, the rules use fuzzy variables; hence, the datasets had to be fuzzified. The labels and fuzzy sets for the diabetic retinopathy problem have been defined from the numerical attributes [2] by medical experts. The numerical attributes for the occupancy problem were fuzzified using Yuan's algorithm [27]. A set of k centers are used to define the membership function for each linguistic label. They start being evenly distributed among the numerical values of the attribute. Then, they are adjusted through an iterative process to reduce the distance between the centers and the numerical values. For each numerical attribute of the occupancy dataset, $k = 5$ linguistic labels have been computed (Very Low, Low, Medium, High and Very High). In both cases, the training algorithm for FRFs that we have used to test the proposed method is explained in a previous work [3].

4.2 Parameter Selection

The main goal of the FRF model update is to improve the sensitivity results on datasets that are highly imbalanced towards the negative class. In such problems, it is hard for the classifiers to detect the positive instances.

In the particular case of the Diabetic Retinopathy disease, doctors want to improve the detection of patients with risk of developing DR, that is, improve the sensitivity of the random forest. Therefore, it is preferred to misclassify non-DR patients as having the risk to develop the disease (False Positive), than the other way around (False Negative). This is due to the very bad consequences of not detecting DR on

Table 3 Diabetic retinopathy sensitivity results

Method	Vote	Update	V0	V1	V2	V3	V4
N	M	N	74.2	77.5	80.4	81.7	–
N	M	Y	74.2	77.5	79.2	80	–
N	W	N	74.6	77.5	81.7	82.1	–
N	W	Y	74.2	77.5	79.2	79.2	–
E	M	N	74.2	78.3	80.8	84.6	85.4
E	M	Y	74.2	78.3	81.2	86.7	87.1
E	W	N	74.6	77.1	81.7	84.6	87.5
E	W	Y	74.2	78.3	82.9	85.8	86.3
A	M	N	74.2	78.3	79.6	85.8	87.9
A	M	Y	74.2	78.3	81.7	87.5	90
A	W	N	74.6	77.1	81.7	83.8	88.7
A	W	Y	74.2	78.3	83.3	86.7	**91.7**

Table 4 Diabetic retinopathy specificity results

Method	Vote	Update	V0	V1	V2	V3	V4
N	M	N	81.7	78.7	78.4	78.2	–
N	M	Y	81.7	78.7	78.1	77.5	–
N	W	N	81.5	78.6	79.4	78.6	–
N	W	Y	81.7	78.6	78	77.4	–
E	M	N	81.7	79.4	81.2	84.2	86.6
E	M	Y	81.7	79.4	81.2	83.7	87.7
E	W	N	81.5	79.1	80.9	83.9	86.1
E	W	Y	81.7	79.4	83	87.5	**91.5**
A	M	N	81.7	79.4	80.8	83	83.2
A	M	Y	81.7	79.4	80.4	83	83.5
A	W	N	81.5	79.1	81.1	82.9	84.2
A	W	Y	81.7	79.4	82.3	84.7	84.4

Table 5 Occupancy sensitivity results

Method	Vote	Update	V0	V1	V2	V3	V4	V5	V6
N	M	N	74.7	74.7	75.9	76.2	76.5	76.5	76.5
N	M	Y	74.7	74.7	76.2	76.2	76.5	76.2	76.2
N	W	N	74.7	74.7	77.1	76.2	76.8	76.5	76.5
N	W	Y	74.7	74.7	76.8	76.2	76.5	76.2	76.2
E	M	N	74.7	75.9	75.9	78	78.9	81.5	88.4
E	M	Y	74.7	75.9	76.2	83.6	86.6	84.5	83.3
E	W	N	74.7	75.9	76.5	76.8	80.1	82.1	87.5
E	W	Y	74.7	75.9	76.5	82.7	86	87.2	85.4
A	M	N	74.7	75.9	76.8	79.2	83	88.7	82.7
A	M	Y	74.7	75.9	76.8	84.8	89.3	86.9	86.9
A	W	N	74.7	75.9	76.8	78.6	80.7	82.7	**89.6**
A	W	Y	74.7	75.9	77.7	85.1	85.7	92.9	86.9

Table 6 Occupancy specificity results

Method	Vote	Update	V0	V1	V2	V3	V4	V5	V6
N	M	N	87.6	85.5	84.6	84.4	85.5	83.9	83.8
N	M	Y	87.6	85.5	84.6	84.4	83.8	84.4	84.1
N	W	N	87.6	85.6	84.5	84.3	85.4	83.8	83.8
N	W	Y	87.6	85.6	84.5	84.3	85.4	84.3	84.4
E	M	N	87.6	85.4	84.8	87.5	90.4	88.9	83.6
E	M	Y	87.6	85.4	84.6	88.7	88.2	88	91.3
E	W	N	87.6	86.9	84.6	86.5	87.6	90.2	85.8
E	W	Y	87.6	87.1	87.4	86.2	91.2	83.8	**92.3**
A	M	N	87.6	85.4	86.7	85.8	81.1	78.9	82.1
A	M	Y	87.6	85.4	84.3	82.1	85.2	84.5	83.9
A	W	N	87.6	86.9	84.4	85.8	83	82.4	75.9
A	W	Y	87.6	87.1	86.7	88.4	89	84.1	84.6

time, which produces a degradation of the vision that may even cause total blindness.

The update method has two parameters: the percentage of trees changed at each iteration, p, and the balancing factor in the calculation of the weighted balanced accuracy, α. After performing an empirical experimentation [28], the percentage was fixed to $p = 10\%$, and the weight in the balanced accuracy to $\alpha = 2/3$. The experiments showed that for $p < 10\%$ the changes on the model were not significant whereas, for higher values, the model suffers too many changes and becomes highly unstable. Regarding the parameter α, the value 2/3 showed a good trade-off to prioritize the improvement of the sensitivity performance without worsening the specificity.

The same parameters were used for the occupancy dataset. Again, in this problem the minority class is the positive one, and sensitivity is the main target, as the goal is to detect if there are people in the room or not.

4.3 Results

This section presents the results of the experimentation with the two datasets. We have tested the different configurations of the method, which include:

- The data used in the updating method: *NoError* (*N*), *ErrorLT* (*E*) or *AllData &ErrorLT* (*A*);
- The voting method: *majority* (*M*) or *weighted* (*W*) voting,
- Whether the metrics are updated at each iteration (*Y*) or not (*N*).

This leads to 12 possible configurations. The obtained sensitivity and specificity results from the Diabetic Retinopathy dataset can be seen in Tables 3 and 4 respectively. The results obtained from the Occupancy dataset can be seen in Tables 5 and 6.

The columns Vx denote the different new validation sets of data received in each iteration. Note that the *NoError* method in the DR dataset has one iteration less than the other configurations. This is due to the fact that this method has less samples because its data are not expanded using the error samples from previous iterations.

A first analysis can be performed on the overall results. The best ones on each table are marked in bold. *NoError* is the method obtaining the worse improvements in both sensitivity and specificity. The highest improvements in sensitivity and specificity, in both datasets, are obtained by the *AllData &ErrorLT* and *ErrorLT* methods respectively.

It can be observed that accumulating the data received in the previous iterations (in the *AllData &ErrorLT* method), we can further improve the sensitivity, keeping a reasonably good level of specificity. In contrast, the *ErrorLT* method does not improve the sensitivity as much, although it has an improvement on the specificity.

To further analyze the results, they are summarized below in three different tables. Each one focuses on one of the features of the method: the update method (*NoError*, *ErrorLT* or *AllData &ErrorLT*), the voting method (*majority voting* or *weighted voting*) and whether metrics are updated or not. In each table there is the difference between the initial sensitivity and specificity values and the final values in the last iteration of the method. There are two values in each cell. The left one is the difference between the worst result and the initial value. The right one is the difference between the best result and the initial value. In both cases, the difference is a percentage, and it is computed flooring the results. This is performed because the weighted voting has slightly different initial results. This range of values represents the minimum and maximum change on the indicator with respect to the base original model. By analyzing the range of variation obtained on each of the methods, an overall view of how each parameter affects the quality of the results can be seen.

In Table 7, there are the results of the different updating versions. On the DR dataset, we can see that the *NoError* method improves the sensitivity values at the cost of slightly decreasing the specificity ones. The *ErrorLT* and *AllData &ErrorLT* methods, in contrast, increase both the sensitivity and specificity values. On the Occupancy dataset, we can see in all methods improvements on the sensitivity, but a decrease on the specificity values. Better results of the sensitivity over the specificity were expected because of the prioritization of the sensitivity in the balanced accuracy calculation. In both datasets, the highest increase of the sensitivity values was achieved by the *AllData &ErrorLT*

Table 7 Method summarized improvement results

Dataset	Method	Sensitivity (%)	Specificity (%)
Diabetic retinopathy	NoError	[6, 8]	[−4, −3]
	ErrorLT	[11, 13]	[5, 10]
	AllData &ErrorLT	[13, 17]	[2, 3]
Occupancy	NoError	[2]	[−4, −3]
	ErrorLT	[9, 14]	[−4, 5]
	AllData &ErrorLT	[8, 15]	[−12, −3]

Table 8 Vote method summarized improvement results

Dataset	Vote	Sensitivity (%)	Specificity (%)
Diabetic retinopathy	Majority	[6, 16]	[−4, 6]
	Weighted	[5, 17]	[−4, 10]
Occupancy	Majority	[2, 14]	[−5, 4]
	Weighted	[2, 15]	[−12, 5]

Table 9 Update metrics summarized improvement results

Dataset	Update	Sensitivity (%)	Specificity (%)
Diabetic retinopathy	No	[7, 14]	[−3, 5]
	Yes	[5, 17]	[−4, 10]
Occupancy	No	[2, 15]	[−12, −2]
	Yes	[2, 12]	[−4, 5]

Table 10 BA and WBA final results

Method	Vote	Update	BA RD	WBA RD	BA OC	WBA OC
N	M	N	79.95	80.53	80.15	78.93
N	M	Y	78.75	79.17	80.15	78.83
N	W	N	80.35	80.93	80.15	78.93
N	W	Y	78.3	78.6	80.3	78.93
E	M	N	86	85.8	86	86.8
E	M	Y	87.4	87.3	87.3	85.97
E	W	N	86.8	87.03	86.65	86.93
E	W	Y	88.9	88.03	88.85	87.7
A	M	N	85.55	86.33	82.4	82.5
A	M	Y	86.75	87.83	85.4	85.9
A	W	N	86.45	87.2	82.75	85.03
A	W	Y	88.05	89.27	85.75	86.13

Initial BA RD is 78 and WBA RD is 76.8; initial BA OC is 81.15 and WBA OC is 79

method. This is at the cost of having lower improvements on the specificity values when compared with the *ErrorLT* method. The lowest increase in the sensitivity is in the *NoError* method. It also does not have better results in the specificity. The *ErrorLT* method has intermediate results. Sensitivity values are increased, and specificity increases and slightly decreases on the DR and OC datasets respectively. We can conclude that the use of error samples on the method gives better results than not using them. Moreover, *AllData &ErrorLT* is able to give better sensitivity results than *ErrorLT* at the cost of a worse specificity.

The results in Table 8 summarize the results of applying either majority or weighted voting in the FRF. It can be seen that the use of weighted voting improves the best result in the majority voting tests, but it also decreases the worse value in majority voting tests. After checking the results, the *NoError* method is the one which gets worse results using weighted voting, whereas *ErrorLT* and *AllData &ErrorLT* get better results using it. According to these results, using the error data is also beneficial when computing the weights of each FDT.

Table 9 summarizes the influence of updating the metrics in each iteration. Updating them keeps similar values

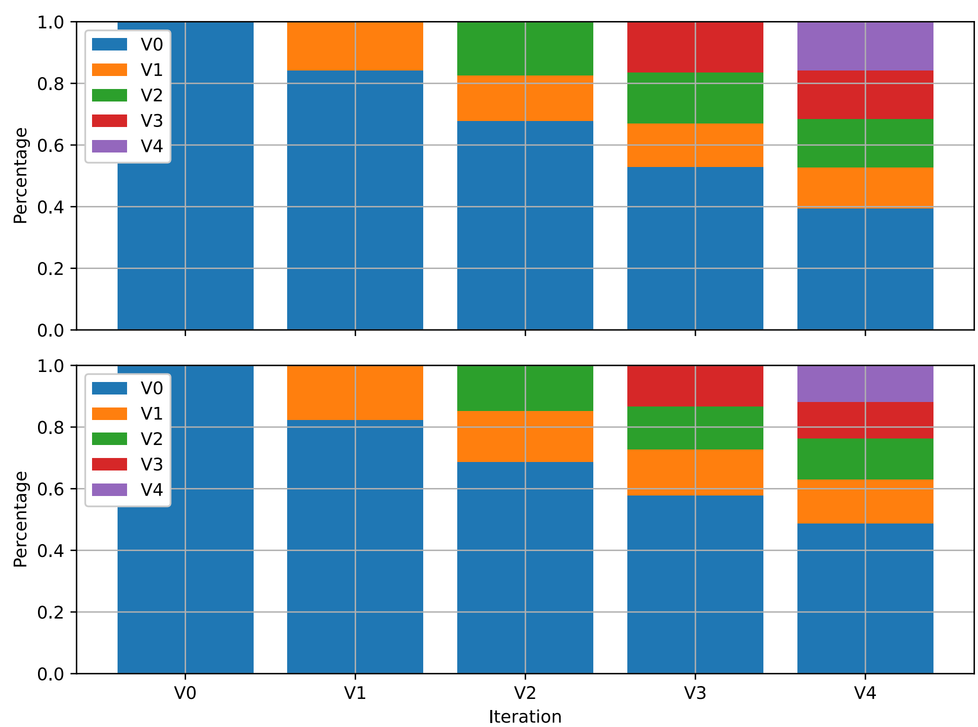

Fig. 2 Evolution of trees in the DR dataset. Update metrics on top, no update below

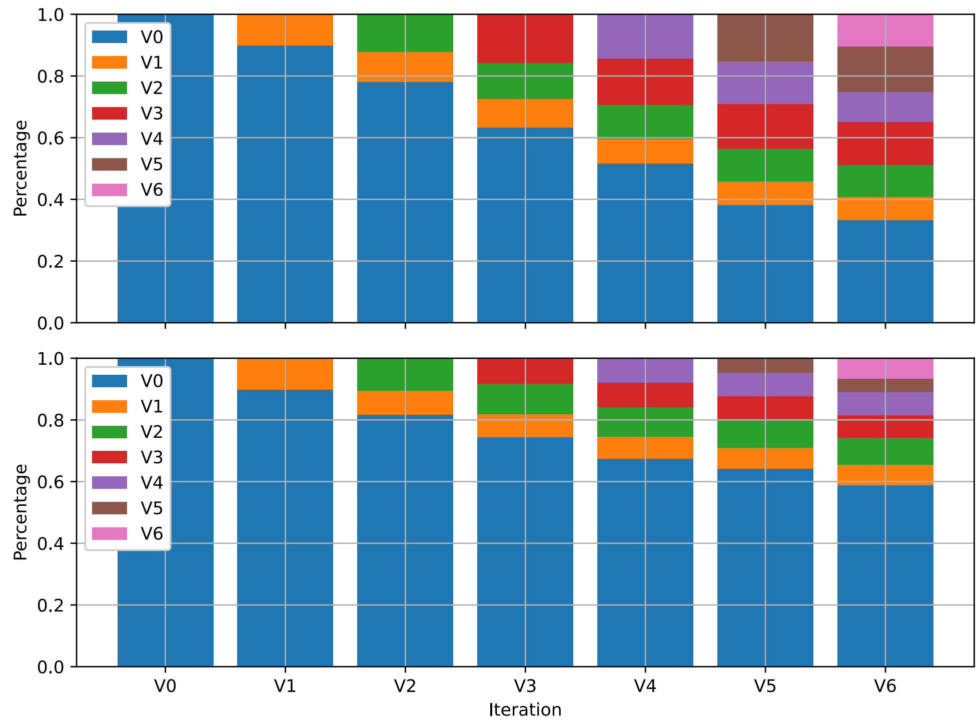

Fig. 3 Evolution of trees in the Occupancy dataset. Update metrics on top, no update below

on the sensitivity, whereas it improves the specificity ones. This update process seems to depend on the other parameters of the method. Observing all the obtained results, if a test configuration not updating the metrics has good results, the update method is able to improve them even more. The opposite also occurs, so when the results are not that good, the updated metrics worsens them.

Finally, on Table 10 we can see the difference in the results of the last iteration between using Balanced Accuracy (BA) or Weighted Balanced Accuracy (WBA) on all the tested configurations. The WBA is computed with a weighting factor $\alpha = 2/3$ as in the proposed method. They can be compared to the BA and WBA of the base model. In the Diabetic Retinopathy dataset, BA is around 78, and WBA is around 76.8, depending on the results of the weighted voting. In the Occupancy dataset, the BA is 81.15, and the WBA is 79.

As expected, *NoError* has the worst results, which are really similar to the base ones in both BA and WBA. When comparing *ErrorLT* and *AllData &ErrorLT*, the final results of both methods are quite similar in terms of the BA. But, as seen in the previous tables, they balance differently the sensitivity and the specificity. *AllData &ErrorLT* has a greater increase in sensitivity than *ErrorLT*, but the increase in specificity is not as high. This is also shown in the WBA, which is greater than its corresponding BA in the *AllData &ErrorLT*, whereas in *ErrorLT* it maintains similar values.

4.4 In-Depth Analysis of the Results

To understand how the proposed method modifies the FRF model, a more detailed analysis of the trees has been performed. The selected configuration for this study is the *AllData &ErrorLT* method and *weighted* voting, because it is the configuration that makes more changes to the FRF during its execution. The analysis includes the cases of updating and not updating the metrics after each iteration, because the update metrics setting produces even more changes to the FRF.

The first analysis, which we can see in Figs. 2 and 3, shows the percentage of trees generated in each iteration that belong to the updated FRF. It can be seen that updating the metrics results in replacing more FDTs from previous iterations than without updating. When there is no update, in the last iteration a 50% and 60% of the trees are kept from the base model in the diabetic retinopathy and the occupancy datasets respectively. This percentage is lowered to 40% and 30% when metrics are updated. As a consequence of this difference, more trees generated in subsequent iterations are present in the last iteration. Despite this difference in both configurations, in general it does not occur that trees are added on one iteration and removed in the following one. That means the trees being incorporated in the FRF are in fact better trees than the removed ones.

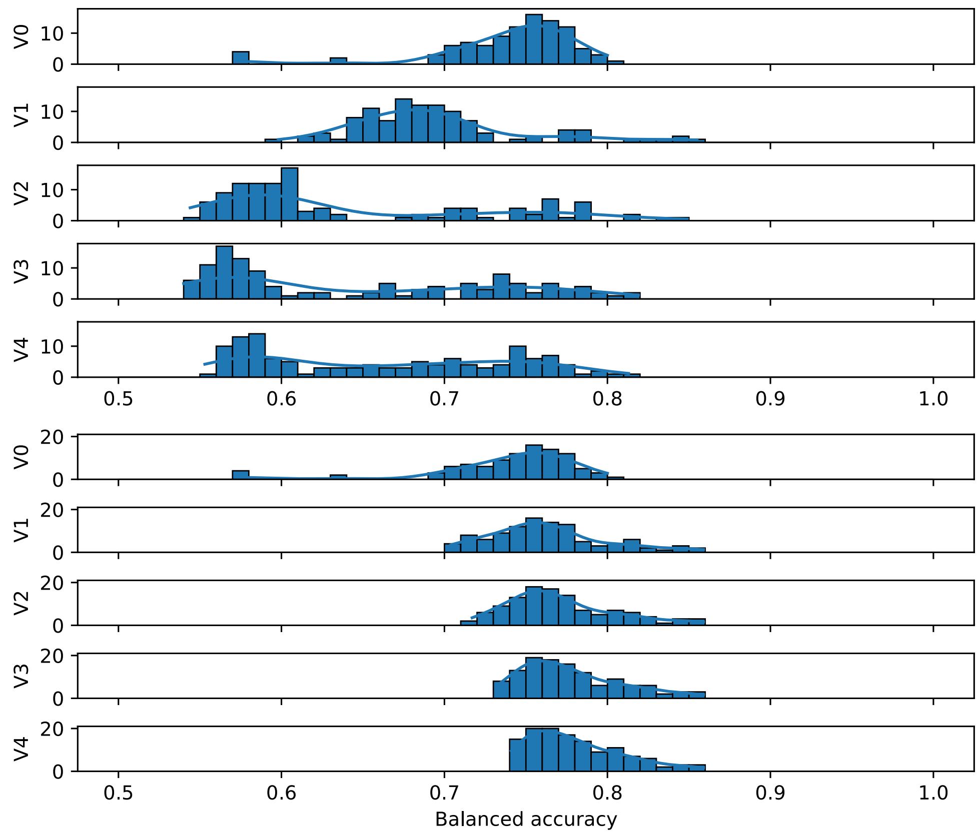

Fig. 4 Histogram of the balanced accuracy of trees in the DR dataset. Update metrics on top, no update below

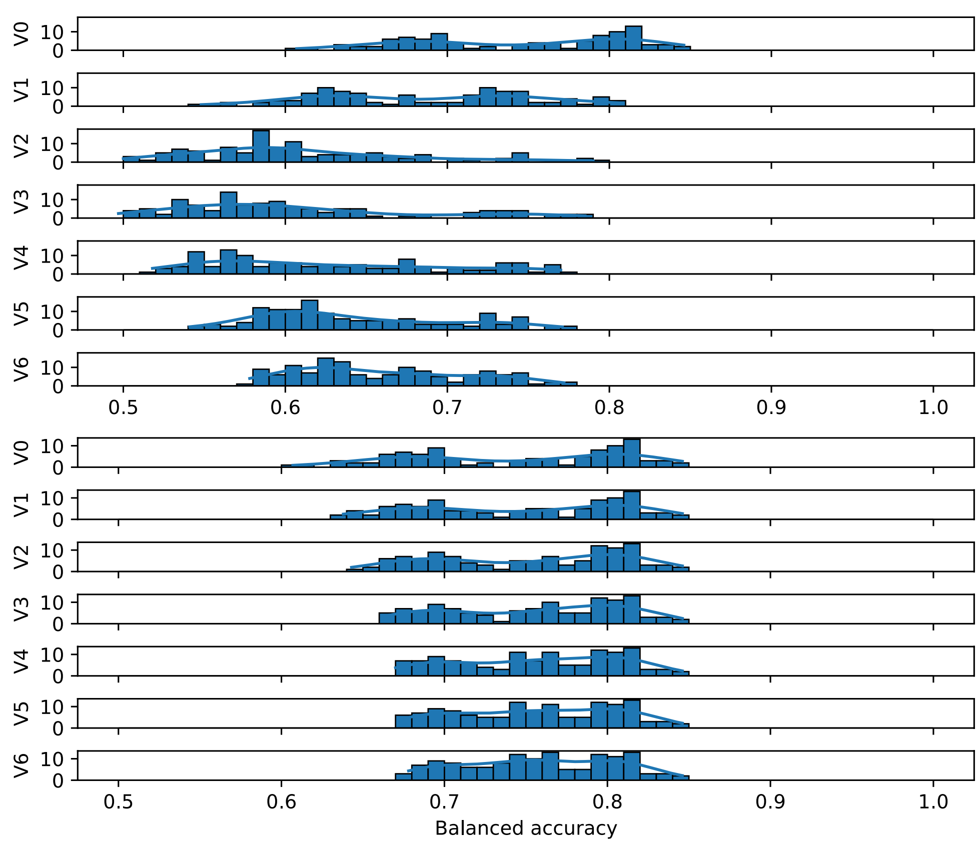

Fig. 5 Histogram of the balanced accuracy of trees in the Occupancy dataset. Update metrics on top, no update below

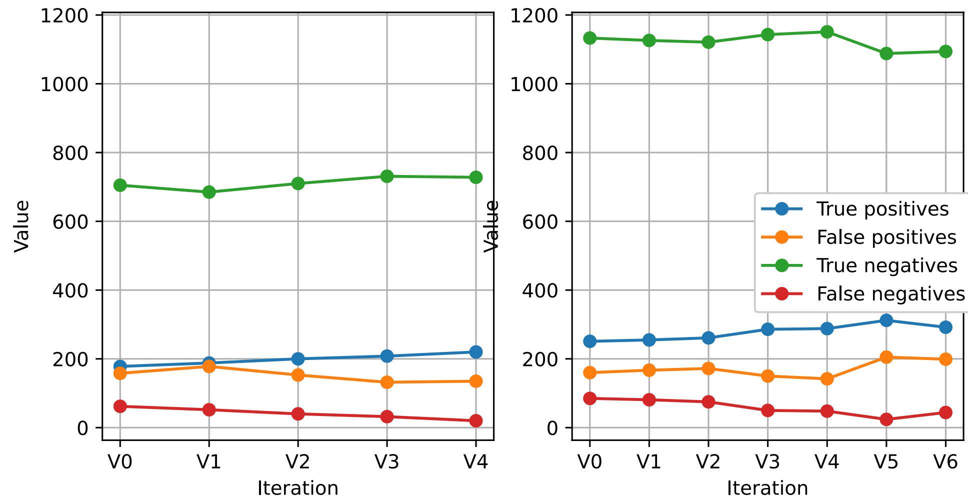

Fig. 6 Confusion matrix on the test set. Diabetic retinopathy (left) and occupancy (right)

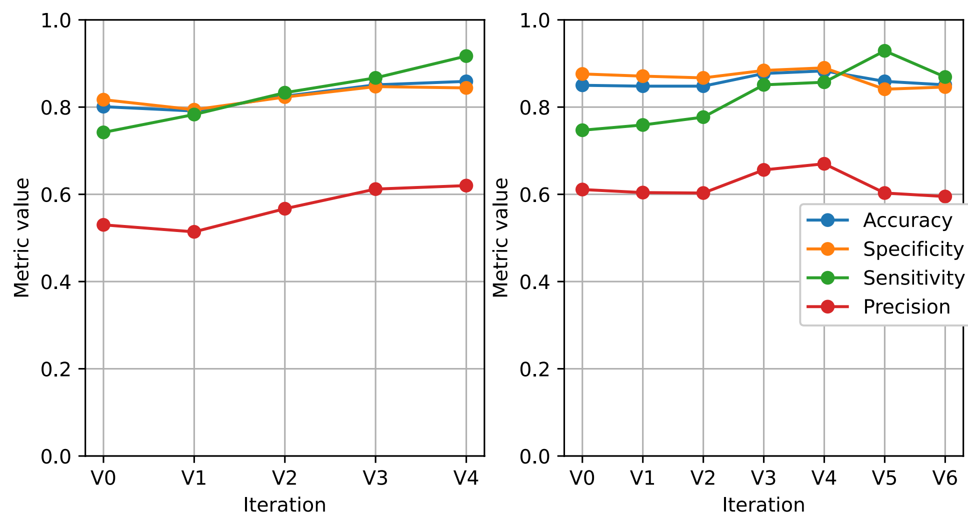

Fig. 7 Metrics on the test set. Diabetic retinopathy (left) and occupancy (right)

Table 11 Results with original, iterative and extended datasets

Dataset	Sensitivity	Specificity	F1-score
Base DR	74.6	81.5	61.8
Extended DR	79.2	78.6	61.8
Iterative DR	91.7	84.4	74
Base OC	74.7	87.6	67.2
Extended OC	75	85.7	65.2
Iterative OC	86.9	84.6	70.6

The second analysis consists on generating a histogram for each iteration. In each of them, we are plotting the balanced accuracy of each tree on the updated FRF. It can be seen in Figs. 4 and 5. When the balanced accuracy is not updated, the initial value computed using the out-of-bag samples is maintained during all iterations. This is why the values increase at each of them, because the worst trees on the lower values are being removed, and trees with higher values are being added.

In contrast, when the weights are updated, all the values are recomputed at each iteration. At first sight it would seem their results are worse, because the values start decreasing. But considering they are recomputed using new data, the fact they start increasing after some iterations proves they are better. Being able to increase the balanced accuracy when using different data each iteration, means the trees generalize better, which ends leading to better results.

The following analysis only includes the tests updating the metrics, as the final results are better than not updating them. In the third analysis, which can be seen in Fig. 6, there are the confusion matrix values obtained on the test set after each iteration. In the diabetic retinopathy dataset, the true positives and true negatives increase, and the false positives and false negatives decrease. Moreover, the changes are gradual during all the update iterations. In the occupancy dataset, the results are similar, with the exception that the true negatives decrease and the false positives increase. Even though these results are not as good as the ones obtained on the diabetic retinopathy dataset, the detection of positive samples has improved, as desired.

The last analysis, illustrated on Fig. 7, shows the evolution of the accuracy, specificity, sensitivity and precision of the updated model in the test set after each update iteration. As expected looking at the confusion matrix results, the metrics on the diabetic retinopathy dataset gradually improve during all iterations. Moreover, the sensitivity is the metric which increases the most, as desired. The occupancy results could also be expected. The sensitivity gradually improves, and the specificity ends slightly decreasing. Even though it is not as desired as improving both metrics, it is still desired for our use case.

The final test shown in Table 11 compares three models. The initial one uses only the first dataset, called Base. A second dataset contains the base training data and also all the validation datasets together. This Extended dataset is used to train a unique model from scratch. The third one is the proposed iterative algorithm. To make a fair comparison with the iterative update method, the weighted voting is also used. And because the last iteration of the update process has more than the 100 trees of the Base model, the Extended datasets are trained with the same number of FDTs, 127 in the diabetic retinopathy dataset and 136 in the occupancy dataset.

The results in Table 11 show that the use of the extended training data gives a better sensitivity and slightly worse specificity than the base models. In contrast, the iterative update method increases in great measure the sensitivity values, whereas the specificity also increases in the diabetic retinopathy dataset, and slightly decreases in the occupancy dataset.

Even though the Extended tests use all the available data and additional FDTs compared to the Base tests, their results are similar, as their F1-scores show. In contrast, the F1-score of the proposed iterative method is greater. Because of the use of the same amount of data and FDTs than the Extended tests, we can say that the proposed method is able to improve the performance of the FRF model.

5 Conclusions and Future Work

The method presented in this paper is able to update a Fuzzy Random Forest in an iterative manner. It allows using newly incoming data to improve the Fuzzy Random Forest model, without having to retrain it from scratch.

It has been tested using data from two different domains: the assessment of the risk of developing diabetic retinopathy, and the occupancy of an office room. Both of them are highly imbalanced towards the negative class, and the base results show a better detection of this class (higher specificity than sensitivity). Moreover, the diabetic retinopathy problem is also inherently ambiguous, given that very similar samples can belong to different classes.

When tested on both datasets, the method has proven to improve the detection of the positive class. Moreover, the detection of the negative class has maintained similar accuracy values, or they have even been improved. The method seems suitable to improve a Fuzzy Random Forest model when new data are available.

In the diabetic retinopathy case, it has been tested using real data of diabetic patients. The proposed method leads to improvements on the assessment of diabetic retinopathy risk, which can help to avoid unnecessary screenings on patients and to reduce the workload of the ophthalmologists. The available resources can also be distributed among the patients, focusing on the ones that really need them.

As future work, we plan to test the proposed method on other domains. We want to study how the use of different metrics affects the results. We would like to consider other domains with different configurations (balanced datasets, using non-fuzzy datasets or other evaluation metrics).

We would also like to add some control mechanisms to check the update process. It would allow to discard updates if they are not improving the Fuzzy Random Forest as expected, or tune them in order to obtain even better results. We should also control the number of samples added in the *AllData &ErrorLT* method, to avoid accumulating too much data in an iteration. After a certain number of iterations, the accumulation should be stopped and a reset made to the set of accumulated data. Even it may be application dependent, we would like to study how to find a suitable number of iterations and a way to discard the less useful ones. This analysis would also let us study the convergence of the method after some iterations.

Author Contributions JPF: conceptualization, methodology, data curation, software, validation, writing—original draft, writing—review and editing. AV: funding acquisition, conceptualization, methodology, validation, writing—original draft, writing—review and editing. AM: conceptualization, methodology, validation, writing—review and editing. PRA: funding acquisition, data curation, validation, writing—review and editing.

Funding This work has been funded by the research projects PI21/00064 and PI18/00169 from Instituto de Salud Carlos III & FEDER funds. The University Rovira i Virgili also supports this work with project 2020PFR-B2-61. The first author has a pre-doctoral FI grant (2021 FI_B 00139) from Secretaria d'Universitats i Recerca de la Generalitat de Catalunya i del Fons Social Europeu.

Availability of data and materials The Diabetic Retinopathy dataset is private and belongs to Hospital Sant Joan de Reus. The Occupancy dataset was obtained from the UCI Machine Learning Repository: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+.

Declarations

Conflict of interest The authors declare no conflict of interests.

Ethics approval and consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., Fernández-Delgado, A.: Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. **15**, 3133–3181 (2014)
2. Romero-Aroca, P., Valls, A., Moreno, A., Sagarra-Alamo, R., Basora-Gallisa, J., Saleh, E., Baget-Bernaldiz, M., Puig, D.: A clinical decision support system for diabetic retinopathy screening: creating a clinical support application. Telemed. e-Health. **25**(1), 31–40 (2019). https://doi.org/10.1089/tmj.2017.0282
3. Saleh, E., Valls, A., Moreno, A., Romero-Aroca, P., Torra, V., Bustince, H.: Learning fuzzy measures for aggregation in fuzzy rule-based models, vol. 11144 LNAI. Springer, pp. 114–127 (2018). https://doi.org/10.1007/978-3-030-00202-2_10
4. Candanedo, L.M., Feldheim, V.: Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models. Energy Build **112**, 28–39 (2016). https://doi.org/10.1016/J.ENBUILD.2015.11.071
5. Winham, S.J., Freimuth, R.R., Biernacka, J.M.: A weighted random forests approach to improve predictive performance. Stat Anal Data Min ASA Data Sci J **6**(6), 496–505 (2013). https://doi.org/10.1002/sam.11196
6. El Habib Daho, M., Settouti, N., Lazouni, M.E.A., Chikh, M.E.A.: Weighted vote for trees aggregation in Random Forest. In: International Conference on Multimedia Computing and Systems—Proceedings. IEEE, pp. 438–443 (2014). https://doi.org/10.1109/ICMCS.2014.6911187.
7. Li, H.B., Wang, W., Ding, H.W., Dong, J.: Trees Weighting Random Forest method for classifying high-dimensional noisy data. In: Proceedings—IEEE International Conference on E-Business Engineering, ICEBE 2010, pp. 160–163 (2010). https://doi.org/10.1109/ICEBE.2010.99
8. Dogan, A., Birant, D.: A weighted majority voting ensemble approach for classification. In: UBMK 2019—Proceedings, 4th International Conference on Computer Science and Engineering, pp. 366–371 (2019). https://doi.org/10.1109/UBMK.2019.8907028. IEEE
9. Zhukov, A.V., Sidorov, D.N., Foley, A.M.: Random forest based approach for concept drift handling. In: Communications in Computer and Information Science, vol. 661. Springer, pp. 69–77 (2017). https://doi.org/10.1007/978-3-319-52920-2_7
10. Kim, H., Kim, H., Moon, H., Ahn, H.: A weight-adjusted voting algorithm for ensembles of classifiers. J. Korean Stat. Soc. **40**(4), 437–449 (2011). https://doi.org/10.1016/j.jkss.2011.03.002
11. Yang, C., Yin, X.C.: Diversity-based random forests with sample weight learning. Cogn. Comput. **11**(5), 685–696 (2019). https://doi.org/10.1007/s12559-019-09652-0
12. Zhong, Y., Yang, H., Zhang, Y., Li, P.: Online random forests regression with memories. Knowl.-Based Syst. **201–202**, 106058 (2020). https://doi.org/10.1016/j.knosys.2020.106058
13. Khan, U., Shin, H., Choi, J.P., Kim, M.: Wfdt: weighted fuzzy decision trees for prognosis of breast cancer survivability. In: Proceedings of the 7th Australasian Data Mining Conference-Vol. 87. Citeseer, pp. 141–152 (2008)
14. Zhu, M., Xia, J., Jin, X., Yan, M., Cai, G., Yan, J., Ning, G.: Class weights random forest algorithm for processing class imbalanced medical data. IEEE Access **6**, 4641–4652 (2018). https://doi.org/10.1109/ACCESS.2018.2789428
15. Livieris, I.E., Kanavos, A., Tampakas, V., Pintelas, P.: A weighted voting ensemble self-labeled algorithm for the detection of lung abnormalities from X-rays. Algorithms **12**(3), 64 (2019). https://doi.org/10.3390/A12030064
16. Utkin, L.V., Kovalev, M.S., Meldo, A.A.: A deep forest classifier with weights of class probability distribution subsets. Knowl.-Based Syst. **173**, 15–27 (2019). https://doi.org/10.1016/j.knosys.2019.02.022
17. Gomes, H.M., Barddal, J.P., Enembreck, I., Bifet, A., Enembreck, F.: A survey on ensemble learning for data stream classification. ACM Comput. Surv **50** (2017)
18. Kalles, D., Morris, T.: Efficient incremental induction of decision trees. Mach. Learn. **24**, 231–242 (1996). https://doi.org/10.1007/bf00058613
19. Utgoff, P.E., Berkman, N.C., Clouse, J.A.: Decision tree induction based on efficient tree restructuring. Mach. Learn. **29**, 5–44 (1997). https://doi.org/10.1023/A:1007413323501
20. Guetova, M., Hölldobler, S., Störr, H.P.: Incremental fuzzy decision trees. Lect. Notes Comput. Sci. (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **2479**, 67–81 (2002). https://doi.org/10.1007/3-540-45751-8_5
21. Ichihashi, H., Shirai, T., Nagasaka, K., Miyoshi, T., Bi, A.: Neuro-fuzzy ID3: a method of inducing fuzzy decision trees with linear programming for maximizing entropy and an algebraic method for incremental learning of E. Fuzzy Sets Syst. **81**, 157–167 (1996)
22. Isazadeh, A., Mahan, F., Pedrycz, W.: MFlexDT: multi flexible fuzzy decision tree for data stream classification. Soft. Comput. **20**(9), 3719–3733 (2016). https://doi.org/10.1007/S00500-015-1733-2/FIGURES/12
23. Pecori, R., Ducange, P., Marcelloni, F.: Incremental learning of fuzzy decision trees for streaming data classification, pp. 748–755 (2020). https://doi.org/10.2991/eusflat-19.2019.102

24. Saffari, A., Leistner, C., Santner, J., Godec, M., Bischof, H.: On-line random forests, pp. 1393–1400 (2009). https://doi.org/10.1109/ICCVW.2009.5457447
25. Lakshminarayanan, B., Roy, D.M., Teh, Y.W.: Mondrian forests: efficient online random forests. Adv. Neural. Inf. Process. Syst. **27**, 3140–3148 (2014)
26. Gomes, H.M., Bifet, A., Read, J., Barddal, J.P., Enembreck, F., Pfharinger, B., Holmes, G., Abdessalem, T.: Adaptive random forests for evolving data stream classification. Mach. Learn. **106**, 1469–1495 (2017). https://doi.org/10.1007/s10994-017-5642-8
27. Yuan, Y., Shaw, M.J.: Induction of fuzzy decision trees. Fuzzy Sets Syst. **69**, 125–139 (1995)
28. Pascual-Fontanilles, J., Valls, A., Moreno, A., Romero-Aroca, P.: Iterative update of a random forest classifier for diabetic retinopathy. Front. Artif. Intell. Appl. **339**, 207–216 (2021). https://doi.org/10.3233/FAIA210136