Random Similarity Forests

Anonymous author
Anonymous organization

***Abstract*—The wealth of data being gathered about humans and their surroundings drives new machine learning applications in various fields. Consequently, more and more often, classifiers are trained using not only numerical data but also complex data objects. For example, multi-omics analyses attempt to combine numerical descriptions with distributions, time series data, discrete sequences, and graphs. Such integration of data from different domains requires either omitting some of the data, creating separate models for different formats, or simplifying some of the data to adhere to a shared scale and format, all of which can hinder predictive performance. In this paper, we propose a classification method capable of handling datasets with features of arbitrary data types while retaining each feature's characteristic. The proposed algorithm, called Random Similarity Forest, uses multiple, domain-specific distance measures to combine the predictive performance of Random Forests with the flexibility of Similarity Forests. We show that Random Similarity Forests are on par with Random Forests on numerical data and outperform them on datasets from complex or mixed data domains. Our results highlight the applicability of Random Similarity Forests to noisy, multi-source datasets that are becoming ubiquitous in high-impact life science projects.**

I. Introduction

Over the last decades, machine learning has matured to the point in which researchers are spoiled for choice with classifiers for *scalar* data. There are hundreds of classifiers to choose from and their performance has been tested on thousands of datasets spanning across various domains [1]. Increasingly, however, researchers are faced with datasets of *complex* composition, where numerical features are not explicitly available. In such cases, there are two main options to choose from. Either to transform complex objects into many simple features which can then be used by any traditional feature-based classifier (e.g., Random Forests [2]) or a dedicated one (e.g., CNNs for images [3]) or to rely on a domain-specific distance measure and use distance-based classifiers (e.g., k-NN [4] or Similarity Forests [5]). However, there is a third option that relies on the fact that many complex data types can be broken down into both traditional features (e.g., numerical or categorical) and simpler but still complex substructures for which good distance measures exist. For example, sequences can be decomposed into subsequences, sets of elements, distributions of elements, or distributions of lengths of subsequences. These complex substructures can then be combined together with simple, numerical or categorical, features and form a *mixed* type dataset to better capture the detailed characteristics of each example. Unfortunately, the currently available classification methods inherently enable only one of these two types of analyses. This limitation may prohibit us from using the full potential hidden in the data, as it forces us to either reduce all complex structures into simple features (most probably with information loss) or process all features with a single, complex distance measure (which is prone to the curse of dimensionality).

Consider a scenario of building a tumor genome classifier predicting patients responding to a given treatment based on whole-genome sequencing (WGS) [6]. Although, theoretically, a distance-based approach can be used in this scenario, it is unlikely to produce good results because of the curse of dimensionality and computational infeasibility (as the reference human genome is ~3.2 billion base pairs long). Moreover, this approach also neglects decades of research in the field of oncology and genomics, which provide a wide array of potentially useful biomarkers. A more natural approach in this scenario is to use a feature-based classifier that can make use of existing biomarkers as well as explore new ones. Currently however, this approach necessitates a drastic decomposition of the data into a fixed set of scalar features. At the same time, the information in the human genome can be decomposed into many intermediate structures carrying significantly more information than their scalar counterparts. Examples of such structures include variant distributions, sequential patterns, gene sets, sequences of sets, interaction graphs. All of these structures have multiple distance measures available and could be used in conjunction with existing numerical biomarkers as a single dataset (as exemplified in Fig. 1) to produce more informed predictions. However, to the best of our knowledge, there does not exist any classifier capable of handling data with mixed, arbitrarily defined types of features.

F_1	F_2	F_3	F_4	F_5	F_6	F_7	y
2	0.75				GGGC		−
4	0.53				ACGTA		+
4	0.78				AGGCG		−
1	1.00				GC		+
6	0.84				CCCTGGT		+

Fig. 1: A schematic illustration of a dataset with *mixed* types, i.e., *numerical* (F_1, F_2) and *complex* ($F_3, \ldots, F_7$) features.

In this paper, we propose a new classifier, called Random Similarity Forests (RSF), capable of handling datasets with mixed, arbitrarily defined feature types. Our approach requires a distance measure to be provided for each feature and works as a blend of Random Forests and Similarity Forests. At each

tree node, a single feature is selected to split the data into two parts. However, since each feature can be defined arbitrarily, the split is not performed according to the feature values directly but indirectly through their pairwise distances. By using a mixture of feature types, Random Similarity Forests inherit the benefits of both feature-based and similarity-based methods. In the paper, we intuitively describe and formally define the proposed classifier. We also perform sensitivity tests with respect to the number of features analyzed in each tree node and the number of trees in the forest. The method is then experimentally evaluated against Random Forests and Similarity Forests on datasets with scalar features, complex objects with a distance measure, and mixed, arbitrarily defined features with distance measures. Our analysis highlights the characteristic properties of each approach and discusses their suitability for different dataset types.

II. Related work

Random forests are arguably one of the most popular classifiers [2] and have been shown to offer the best quality of predictions across a wide range of datasets when compared to other classifiers [1]. However, like many other approaches, their use is limited only to scalar data and, therefore, require feature extraction when dealing with complex objects. Due to this fact, Similarity Forests have been recently proposed as an alternative to Random Forests, especially when regular features are absent but similarities or distances between examples are attainable [5]. Although similar methods for building decision trees based on directions defined with pairs of examples have already existed [7], they still required a regular feature space.

When dealing with distance-based classification, kernel methods are a popular group to consider, especially since there have been studies exploring the relationship between Random Forests and kernel methods [8]. In general, distance-based classification is often applied when dealing with complex data, e.g., for microbiome [9], image [10], [11], or time series classification [12].

A different way of dealing with complex data in classification using distances is through a combination of clustering and distance-based feature extraction. This approach relies on unsupervised clustering of the data and later encoding the clusters as features, either through aggregate inter- and intra-cluster distances [13] or simply by encoding the distances to each cluster center as separate features [14].

Recently, a different combination of feature-based and distance-based methods has been proposed by Liang *et al.* [15], where the authors first use Random Forests to select the most important features according to their impurity-based importance and later use these features in a k-Nearest Neighbors classifier with dynamically selected distance measures.

Although the described existing approaches create forests or calculate distances between objects, they either require scalar features or are based on a single distance calculated for entire examples. Similarly, integrative approaches from biomedical studies that combine data from different domains either require a common feature representation, create separate data type specific models, or perform data re-scaling and simplification [16]. To the best of our knowledge, Random Similarity Forests are the only single-classifier model capable of dealing with datasets with a mixture of arbitrarily defined features without any data transformations.

III. Random Similarity Forest

A. Preliminaries

Given a dataset of *training examples* $\mathcal{X} = \{\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}\}$ and their corresponding *class labels* $\boldsymbol{y} = \{y_1, y_2, \ldots, y_n\}$, the task of a *classifier* is to predict the class label $\hat{y}$ of each *unlabeled example* $\hat{\boldsymbol{x}}$ (i.e., an example for which the class is unknown). Every example $\boldsymbol{x_i}$ is a list $(x_{i1}, \ldots, x_{ij}, \ldots, x_{ip})$ of the same length p, where each position j holds a value a given example has for a particular *feature* $F_1, F_2, \ldots, F_p \in \mathcal{F}$. We also denote all values of $\mathcal{X}$ across a given feature F_j by $\boldsymbol{x_{.j}}$. Each class label y_i falls into one of several categorical class values $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$. This is the standard definition of a classification problem. In our case, each feature can be from a different, arbitrary data domain. Additionally, every feature $F_1, F_2, \ldots, F_p \in \mathcal{F}$ has an associated distance measure $\delta_1, \delta_2, \ldots, \delta_p \in \Delta$. Therefore, the distance between any two examples $\boldsymbol{x_i}$, $\boldsymbol{x_j}$ for a given feature F_l is equal to $\delta_l(x_{il}, x_{jl})$, or $\delta_l(\boldsymbol{x_i}, \boldsymbol{x_j})$ for short.

B. The Algorithm

Just like Random Forests and Similarity Forests, Random Similarity Forests rely on bagging of a user-defined number (*max_trees*) of single-tree classifiers. A single Random Similarity Tree is constructed in a top-down fashion by recursively splitting the examples in each node into two exclusive subsets. The recursion stops when a given node contains only examples from a single class or when one of the early-stopping conditions is met (discussed in Section III-D). Each split is calculated using a similarity-based 1-dimensional projection of the examples in a given node. First, a subset of *max_features* candidate features is selected, and afterward, *max_pairs* pairs of examples are picked at random for each feature, where *max_features* and *max_pairs* are user-defined parameters. The examples are selected so that they come from different classes and have a different value on a given feature.

For a given pair of examples $\boldsymbol{x_p}$, $\boldsymbol{x_q}$, the distance between all other examples $\boldsymbol{x_i} \in \mathcal{X}_{node} \setminus \{\boldsymbol{x_p}, \boldsymbol{x_q}\}$ and the two selected examples is evaluated for each selected feature F_l in order to create a projection $\mathcal{P}(\boldsymbol{x_i})$ of each example $\boldsymbol{x_i}$ into a direction defined by $\boldsymbol{x_p}$ and $\boldsymbol{x_q}$. As shown by Sathe and Aggarwal [5], for the purposes of constructing a split, this projection can be approximated with $\mathcal{P}(\boldsymbol{x_i}) \propto \delta_l(\boldsymbol{x_q}, \boldsymbol{x_i}) - \delta_l(\boldsymbol{x_p}, \boldsymbol{x_i})$, as this approximation preserves the order of the original projection and the order is the only information necessary to construct a split. Therefore, for each pair of examples $\boldsymbol{x_p}$, $\boldsymbol{x_q}$, all remaining examples $\boldsymbol{x_i}$ in a given node are ordered by $\delta_l(\boldsymbol{x_q}, \boldsymbol{x_i}) - \delta_l(\boldsymbol{x_p}, \boldsymbol{x_i})$ along a given feature F_l.

This projection approximation can be viewed as a dynamic feature. After the projection has been computed, it can be used

to perform a split just like any typical numerical feature would be used in a regular decision tree. The split point is selected as the point which minimizes the weighted average of the Gini index of the children nodes. For a given node N_i, the Gini index is defined as

$$G(N_i) = 1 - \sum_{c \in \mathcal{C}} p_c^2,$$

where p_c is the fraction of examples of class c in a given node. Therefore, given two nodes N_i, N_j consisting of n_i, n_j examples, respectively, the weighted Gini index is calculated as:

$$G(N_i, N_j) = \frac{n_i G(N_i) + n_j G(N_j)}{n_i + n_j}. \quad (1)$$

Here, we opted for the Gini-index-based splitting strategy used also in Similarity Forests, however, it is important to note that after the projection is done we are in fact in possession of a dynamically calculated numerical feature. Therefore, other measures, such as entropy, can also be used.

Given the order defined with the projection and an impurity measure, all possible splits along a given feature are evaluated and the split minimizing the impurity is selected. If there is more than one split point with the same, minimum impurity, the one closest to the center of the projection is selected (i.e., the one producing the most balanced split). After performing this procedure for all selected features in a given node, the feature producing the best split is selected and saved in the node together with the selected random points and the threshold used to perform the split. Subsequently, two new tree nodes are created and the examples falling below the split threshold are placed in one node while the remaining examples are placed in the other node, and the whole procedure is repeated until all leaf nodes are pure or an early-stopping criterion is met.

The above-described process is illustrated in Fig. 2. The pseudocode of training a single Random Similarity Tree is presented in Algorithm 1.

The condition in Line 1 is the stopping condition for recursion at which the algorithm has arrived at a leaf node by either reaching a pure node (i.e., all examples of the same class) or fulfilling an early-stopping condition (discussed in Section III-D). If the stopping condition is not met, the algorithm starts randomly picking a specified number of features with non-zero variance (Lines 5–6). Line 7 selects the class from which to randomly pick the first of the pair of examples as the class with the minimum variance over the feature selected in the previous step. This is to ensure that if all of the examples of one of the classes have the same feature value (zero variance), we will still be able to pick an example with a different value from the second class. Next, in Lines 9–11 we randomly pick a specified number of pairs of examples so that they come from different classes and are of different values. With a pair of selected examples, in Lines 13–14 we perform a distance-based projection which we then use in Lines 15–21 to find the best split point thr. For each unique value on the projection (Line 15), we split the examples into

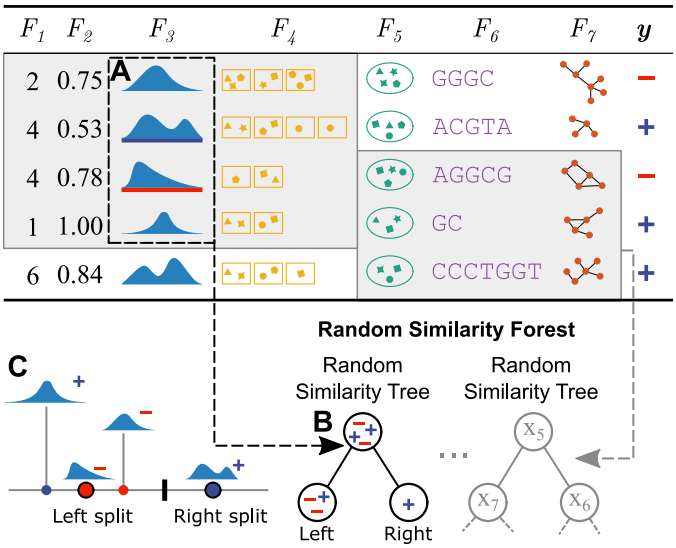

Fig. 2: An example illustrating the training process of Random Similarity Forests. Gray areas symbolize the bootstrap samples and subset of features going into each tree. Feature F_3 (**A**) is selected to perform a split in one of the trees (**B**). The distance-based projection (**C**) for that feature is calculated and the split point yielding the purest split is selected.

Algorithm 1 Random Similarity Tree: $RST(\cdot)$

Require: $\mathcal{X}$: training examples; $\boldsymbol{y}$: examples' class labels; $\boldsymbol{\delta}$: distance measures for each feature; *max_features*: number of features picked randomly at each node; *max_pairs*: number of example pairs picked at random for each feature at each node
Ensure: a trained Random Similarity Tree model

1: **if** $earlyStopping()$ **or** $len(unique(\boldsymbol{y})) = 1$ **then**
2: $setLeaf()$
3: **return** *self*
4: **else**
5: **for** $1..max\_features$ **do**
6: $F_j \leftarrow random(\{F_j \in \mathcal{F} : var(\boldsymbol{x_{.j}}) > 0\})$
7: $c_1 \leftarrow \text{argmin}_{c \in unique(\boldsymbol{y})}\{var(\{x_{ij} : y_i = c\})\}$
8:
9: **for** $1..max\_pairs$ **do**
10: $\boldsymbol{x_p} \leftarrow random(\{\boldsymbol{x_i} : y_i = c_1\})$
11: $\boldsymbol{x_q} \leftarrow random(\{\boldsymbol{x_i} : y_i \neq c_1 \wedge x_{ij} \neq x_{pj}\})$
12:
13: **for** $\boldsymbol{x_i} \in \mathcal{X}$ **do**
14: $P(\boldsymbol{x_i}) \leftarrow \delta_j(\boldsymbol{x_q}, \boldsymbol{x_i}) - \delta_j(\boldsymbol{x_p}, \boldsymbol{x_i})$
15: **for** $thr \in unique(P)$ **do**
16: $\boldsymbol{y_{left}} \leftarrow \{y_i \in \boldsymbol{y} : P(\boldsymbol{x_i}) \leq thr\}$
17: $\boldsymbol{y_{right}} \leftarrow \{y_i \in \boldsymbol{y} : P(\boldsymbol{x_i}) > thr\}$
18: $g \leftarrow G(\boldsymbol{y_{left}}, \boldsymbol{y_{right}})$
19:
20: **if** $(g < best_g)$ **or** $(g = best_g$ **and** $|len(y_{left}) - len(y_{right})| < best_{bal})$ **then**
21: $update\_best(g, thr, F, \boldsymbol{x_p}, \boldsymbol{x_q}, P, bal)$
22:
23: $\mathcal{X}_{left}, \boldsymbol{y_{left}} \leftarrow \{\boldsymbol{x_i} \in \mathcal{X}, y_i \in \boldsymbol{y} : P(\boldsymbol{x_i}) \leq thr\}$
24: $\mathcal{X}_{right}, \boldsymbol{y_{right}} \leftarrow \{\boldsymbol{x_i} \in \mathcal{X}, y_i \in \boldsymbol{y} : P(\boldsymbol{x_i}) > thr\}$
25: $N_{left} \leftarrow RST(\mathcal{X}_{left}, \boldsymbol{y_{left}}, \boldsymbol{\delta}, max\_features, max\_pairs)$
26: $N_{right} \leftarrow RST(\mathcal{X}_{right}, \boldsymbol{y_{right}}, \boldsymbol{\delta}, max\_features, max\_pairs)$
27: **return** *self*

two groups based on this value (Lines 16–17), evaluate the split using the Gini quality defined in Eq. 1 (Line 18), and store all of its parameters in the tree node if the split is better (purer or equally pure but more balanced) than the previous best split (Lines 20–21). Finally, we split the examples in the node using the best parameters and create two new child nodes from the split examples (Lines 23–26).

After all *max_trees* Random Similarity Trees are computed, the trained Random Similarity Forest is ready to make predictions. For each node in each tree, given an unlabeled example $\hat{\boldsymbol{x}}$, the example is projected along the feature and the two examples stored in that node and assigned to one of the child nodes based on the stored split point. Following this procedure, once the example reaches a leaf node, it returns a weighted prediction of the majority class in that node. After predictions from all trees in the forest are made, the example is assigned to the class with the highest weighted average from all single-tree predictions.

C. Computational Complexity

Let us now discuss the computational complexity of Random Similarity Forests (RSF). Unsurprisingly, it is strongly related to the complexity of Similarity Forests (SF), which is $\mathcal{O}(n \cdot \log n)$. However, RSF additionally adds the cost of checking multiple features at each node, so the complexity is raised by that factor to $\mathcal{O}(p \cdot n \cdot \log n)$, which is equivalent to the complexity of Random Forests (RF). However, it is worth noting that in practice the value of the *max_features* parameter for RSF may need to be higher than for RF as it is inherently more random because it randomly picks both features and pairs of examples. Moreover, even though the number of pairs is a parameter, we recommend picking only a single pair for each feature, as suggested by Sathe and Aggrawal [5].

Another cost hidden in the computational complexity is the cost of calculating the distances between examples. It is a very important factor because, depending on the feature type, it can either be negligible or a very costly operation. For simple, numerical features, the distance computation can be usually omitted as for any metric it would produce a projection equivalent to the original feature in terms of the example order. On the other hand, more complex data types may require substantial computation, e.g., edit distance has quadratic complexity for sequential data and is NP-complete for graphs. Therefore, the use of such metrics may be prohibitive for longer sequences or larger graphs.

D. Discussion

Since Random Similarity Forests draw from Random Forests and Similarity Forests, they combine the advantages from both of these methods and possess unique properties:

1) By using feature-oriented similarity metrics to split data points, Random Similarity Forests can be used to classify datasets characterized by numerical as well as complex data features. Random Forests cannot classify complex data objects, whereas Similarity Forests treat entire examples as data objects, hiding the characteristics of individual features.
2) By sampling feature subsets for each node split and by analyzing each feature separately, Random Similarity Forests are robust to noisy datasets with a lot of features. Similarity Forests treat entire examples as data objects, which can hinder their performance on datasets with many irrelevant features.
3) Analyzing similarities based on each feature separately allows Random Similarity Forests to use more than one similarity measure to analyze the same data objects and let the forest decide which measures are the most useful for classification.

In addition to combining the advantages from both Random Forests and Similarity Forests, Random Similarity Forests also inherit some of their limitations which are worth discussing. The first limitation comes from Similarity Forests and it is the fact that it is designed for binary classification problems. However, one can easily adapt it to multi-class classification or regression problems analogously to the method proposed by Czekalski and Morzy for Similarity Forests [17], or simply by using the one-vs-all approach.

Another characteristic worth discussing is the effect of feature variance on node splits. If one would select the pair of examples $(\boldsymbol{x_p}, \boldsymbol{x_q})$ with the same value of feature F_j, then they would produce a completely random ordering of examples for the split (since if $x_{pj} = x_{qj}$ then $\forall \boldsymbol{x_i} : \delta_j(\boldsymbol{x_q}, \boldsymbol{x_i}) - \delta_j(\boldsymbol{x_p}, \boldsymbol{x_i}) = 0$). Similar problems appear in Random Forests when the feature values are the same for examples of different classes, whereas Similarity Forests do not suffer from this particular limitation so often as they always consider each example as a whole. For datasets with many features with low variance, this reduces the effective number of features tested at each node and promotes high-variance features. That is why Random Similarity Forests will perform better on datasets with features with high variance. This limitation can be overcome by tuning the *max_features* hyperparameter, responsible for the number of features tested at each node. As shown by Geurts *et al.* [18], the best value for this parameter highly depends on the number of correlated and irrelevant features: lower values work better with highly correlated features while higher values with irrelevant features. Indeed, as will be shown in Section IV-C, the proposed default value of *max_features* for Random Similarity Forests is $0.5p$, which is higher than $\sqrt{p}$ commonly used for Random Forests [18].

Other issues worth mentioning are the early-stopping conditions (pre-pruning) and pruning (post-pruning). There are many possible options to stop the tree construction process before it produces perfectly pure leaf nodes. These could include, e.g., limiting the maximum depth a tree can reach, specifying the minimum number of elements required to perform a split or the minimum number of elements to form a leaf node. Similarly, there are multiple ways in which a tree can be pruned, with arguably the most popular method being minimal cost complexity pruning [19]. However, pre-and post-pruning methods are not specific to the proposed algorithm or

any other forest algorithm and are out of the main scope of this paper.

IV. Experimental evaluation

In this section, we experimentally evaluate Random Similarity Forests and illustrate their usefulness in real-world scenarios. Since the proposed classifier is related to both Random Forests and Similarity Forests, the nature of the evaluation is highly comparative. Our goal is to showcase the strengths and limitations of our proposal and indicate when its use would be the most beneficial and when it might be preferable to rely on one of the alternatives. First, we compare the classification performance of the three classifiers on ten publicly available datasets with simple numerical features. Next, we compare the three approaches on complex datasets, i.e., datasets consisting of arbitrarily defined objects for which there are no numerical features and only a distance function is defined. Finally, we illustrate the usefulness of our proposal by training the classifiers on datasets consisting of mixtures of complex object-like features and simple numerical features.

A. Experimental Setup

In all of the experiments, we compare Random Similarity Forests (RSF) against Random Forests (RF) and Similarity Forests (SF) using the area under the ROC curve (AUC). We chose AUC since we focus on binary classification, it is skew invariant, assesses the classifiers' ranking abilities, and it has been shown to be statistically consistent and more discriminating than accuracy [20]. To evaluate each approach on each dataset, we rely on 10 repetitions of stratified 2-fold cross-validation. Afterwards, the results undergo a series of statistical tests [21]. We rely on the Friedman statistic with a post-hoc Nemenyi test to distinguish between the compared approaches across all datasets. Afterwards, we check for differences on each individual dataset. First, we verify the normality assumption using the Shapiro-Wilk test. If the assumption is met, we use the ANOVA statistic with a post-hoc Tukey HSD test. If the assumption is not met, we once again resort to the non-parametric Friedman statistic with a post-hoc Nemenyi test [21]. All statistical tests were performed at significance level $\alpha = 0.05$.

The code used to perform the experiments was written in the Python programming language with parts of the implementation of the Random Similarity Forests written in Cython to achieve higher efficiency. For Random Forests, we use the scikit-learn implementation [22], whereas for Similarity Forests, we rely on the implementation described in [17]. Unless stated otherwise, we rely on the default hyperparameters for each classifier, as specified in their implementations. The source code, datasets, reproducible experimental scripts, visualizations, and results are publicly available at <triple-blind review — available upon acceptance>.

B. Datasets

Since our experimental analysis focuses on data with scalar, complex, and mixed types of features, we use three groups of datasets, each corresponding to a given feature type. Additionally, we have a separate set of datasets dedicated to performing sensitivity analyses. The choice of datasets was made a priori and independently of the results obtained with our methods.

Datasets for sensitivity analysis: For sensitivity analysis we use binarized versions of 12 numerical datasets: ten from the UCI Machine Learning repository [23] and two introduced by Breiman [24]; for a detailed description of these datasets see [18]. We chose these datasets as they were used to discuss the impact of different hyperparameter values for Random Forests and Extra Trees [18]. Importantly, these datasets are independent of those used for assessing the predictive performance of RF, SF, and RSF, in order to avoid biasing the comparative analysis.

Scalar data: For the analysis of scalar data, we used ten publicly available datasets with numeric features. The datasets exhibit various conditions in terms of the number of features, learning sample size, and feature variance, and can be accessed through the LIBSVM dataset repository [25]. We used the scaled versions of the datasets, as required by Similarity Forests. Each dataset was additionally preprocessed by removing features containing missing values. Table I presents the main characteristics of each dataset after preprocessing.

Complex data: For the analysis of complex data (i.e., data where each example is an object without any explicitly defined numerical features), we use three synthetic datasets consisting of sequences of sets [26]. Sequences of sets are a good candidate for complex data because they combine set information, which can be easily encoded with numerical features, and sequential information, which is very difficult to encode as scalars but manifests itself through distance measures on whole objects. Therefore, we used a synthetic data generator available at https://gingerbread.shinyapps.io/SequencesOfSetsGenerator/ to create three datasets of sequences of sets: `items`, `lengths`, and `order`. The sequences in the `items` dataset have the same mean length and mean set size, and are differentiated by the distributions of the selected items between the two classes. Since this information is set-specific, it should be possible to capture using a simple bag of words representation. Sequences in the `lengths` dataset are generated with identical set size and item distributions but are differentiated by the lengths between the classes. This information should be much harder for the bag of words representation to pick up. Finally, all the sequences in the `order` dataset are generated from the same length, set size, and item distributions, and are only differentiated by the order of the elements in one of the classes. In this class, the items in each set are generated from a distribution slightly shifted w.r.t. the previous set. The sets in the other class are not ordered. This distinction can only be made by analyzing the sequence as a whole and cannot be detected when analyzing each set in isolation.

Mixed data: For the analysis of mixed data (i.e., data where each example is described by arbitrarily defined features with a distance measure for each feature), we use four real-world datasets. The first two datasets (`chickenpieces`,

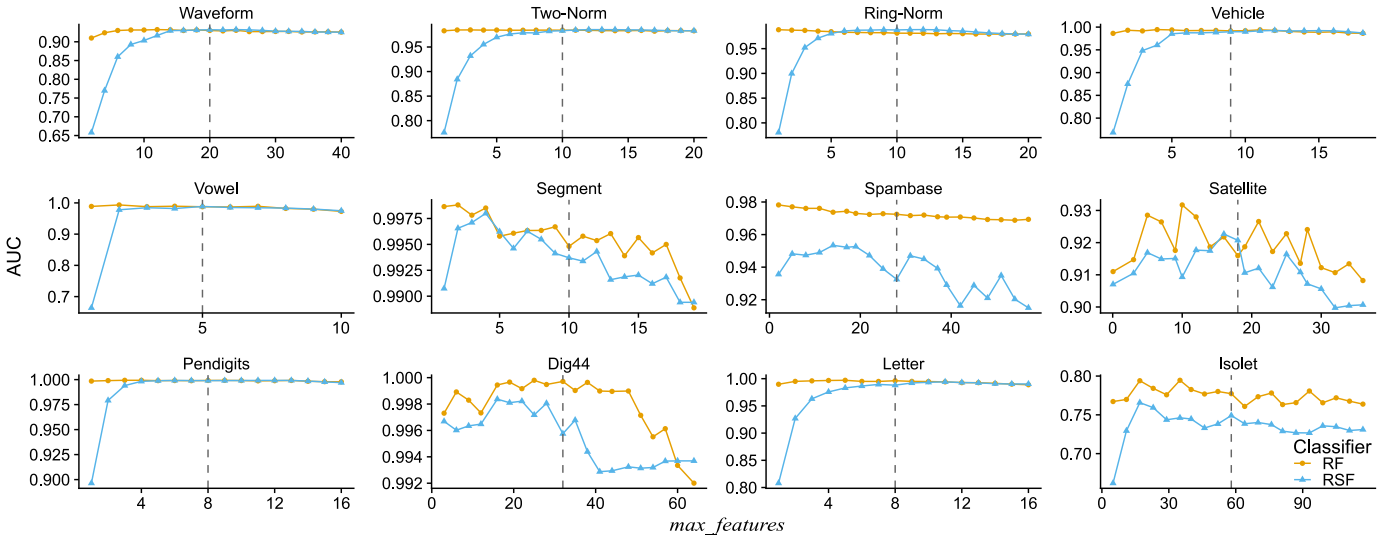

Fig. 3: Evolution of mean AUC of Random Forests (RF) and Random Similarity Forests (RSF) with *max_features* on 12 datasets. Dashed line shows the proposed RSF default value of *max_features*=$0.5p$.

`toolset`) consist of objects for which multiple edit-distance measures have been pre-computed for string representations of shapes. Therefore, each dataset can be viewed as a mixture of various features with pre-computed distances. For a detailed description of these datasets see [27]. The fourth dataset represents tumor samples of ovarian cancer patients obtained using whole genome sequencing, where the goal is to predict the patients' response to treatment. Each sample is characterized by 38 gene amplifications represented as simple, numerical features and 6 distributions of lengths of various structural variants found in the patient's DNA (e.g., insertions).

C. Sensitivity Analysis

Using the 12 datasets proposed in [18], we have analyzed the effect of *max_features* parameter on Random Similarity Forests. The parameter *max_features* denotes the number of potential splits screened at each node during the growth of a Random Similarity Tree. This parameter may be chosen in the interval $[1, \ldots, p]$ and the smaller its value the stronger the randomization of the trees. Figure 3 compares the AUC of Random Forests (RF) and Random Similarity Forests (RSF) for increasing values of *max_features*.

One can notice that for all the datasets the performance of Random Similarity Forests monotonically increases or increases and then declines with increasing *max_features*. As mentioned in Section III-D, this phenomenon is related to the number of correlated and irrelevant features in a dataset: lower values work better with highly correlated features while higher values of *max_features* work better with irrelevant features. This is due to the fact that a higher value of *max_features* leads to a better chance of filtering out the irrelevant variables. These findings are in accordance with [18], were similar trends were noticed for the Extra Trees classifier. It was worth noting, however, that compared to Random Forests, the increase of predictive performance with *max_features* is slower. This stems from the fact that Random Similarity Forests sample pairs of examples and create different projections, thus introducing additional randomness. Therefore, we propose to use a default value of *max_features*=$0.5p$, as opposed to $\sqrt{p}$, which is the commonly used default for Random Forests and Extra Trees [18].

We also analyzed the *max_trees* parameter, responsible for the number of trees in the ensemble. It is well known that for randomization methods the behavior of predictive performance is a monotonically increasing function of *max_trees* [18]. Therefore, the higher the value of *max_trees*, the better the ensembles accuracy. However, for the sake of method comparison, Fig. 4 shows the evolution of mean AUC of RF, SF, and RSF on the Waveform dataset. It can be noticed that all three algorithms have the same speed of convergence. In the remaining experiments, for all three classifiers, we use *max_trees*=100, which is large enough to ensure convergence of the ensemble effect with all our datasets.

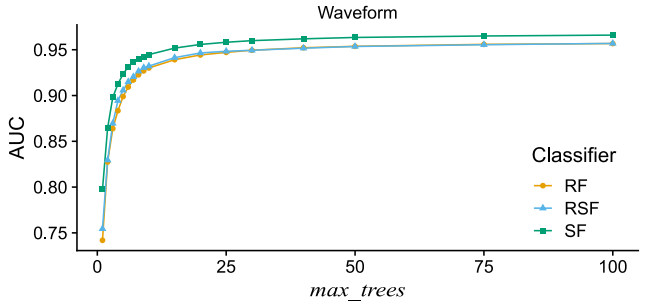

Fig. 4: Evolution of mean AUC of Random Forests (RF) and Random Similarity Forests (RSF) and Similarity Forests (SF) with *max_trees* on the Waveform dataset.

D. Evaluation on Scalar Data

The aim of the first set of experiments is to assess the predictive performance of Random Similarity Forests (RSF) on scalar data. Since scalar features are the natural data domain of Random Forests (RF), the aim is also to verify whether the proposed classifier will be able to match the performance of Random Forests. Since Similarity Forests (SF) are also capable of dealing with scalar data and were reported to produce results on par with or even better than Random Forests [5], we include this approach in this comparison as well. The results of these experiments are presented in Table I and depicted on Fig. 5.

TABLE I: Dataset characteristics and mean AUC performance of Random Forests (RF), Similarity Forests (SF), and Random Similarity Forests (RSF) from 10 × 2-fold cross-validation.

Dataset	Type	#Ex.	#Features	AUC		
				RF	RSF	SF
australian	scalar	690	8	**0.92**$^{\circ}$	0.91	0.92°
breast		683	10	0.99	0.99	**0.99**$^{\bullet}$
diabetes		768	6	**0.80**	0.80	0.80
german		1000	18	**0.73**$^{\circ}$	0.69	0.72°
heart		270	9	0.88	0.88	**0.89**
ionosphere		351	2	0.84	**0.86**$^{\bullet}$	0.84
liver		145	4	**0.77**	0.76	0.76
sonar		208	59	0.89	0.87	**0.91**$^{\circ}$
splice		1000	60	**0.99**$^{\circ}$	0.98°	0.90
svmguide3		1243	21	**0.84**$^{\circ}$	0.84°	0.77
items	complex	400	9–33†	**1.00**$^{\bullet b}$	0.91°	0.84
length		400	5–37†	0.86^{b}	0.91°	**0.91**$^{\circ}$
order		400	7–32†	0.56^{b}	**0.79**$^{\bullet}$	0.74°
chickenpieces	mixed	446	44	-	**0.99**	0.99^{m}
toolset		47	24	-	**0.96**	0.93^{m}
wgsovarian		219	44	0.69$^{\circ s}$	**0.76**$^{\bullet}$	0.61^{s}

$^{\bullet}$ result significantly better than both competitors
$^{\circ}$ result significantly better than one of the competitors
† min–max sequence length; each set contained from 4 to 38 elements
^{b} RF used a bag of words representation of the data
^{m} SF median score achieved by all available distances
^{s} RF and SF used data with distribution features encoded as sums

The results clearly demonstrate that all three approaches are capable of producing high quality results on scalar data. Nevertheless, there are some differences between the reported performance of the approaches. For datasets `australian`, `breast`, `diabetes`, `heart`, and `liver` the results are very close, and either of the approaches (RF, SF, RSF) could be used. On datasets `german` and `sonar`, RSF performs slightly worse than RF and SF, while on dataset `ionosphere` it outperforms both competitors. Interestingly, on datasets `splice` and `svmguide3` RSF produces very similar results to RF, but both approaches significantly outperform SF by a large margin. This outcome is probably due to the fact that although both RSF and SF are distance-based, the former still treats each feature separately, while the latter calculates distances over the whole feature space which, in this case, clearly obfuscates the decision boundary.

The main goal of this experiment was to check if Random Similarity Forests can compete with Random Forests in their natural environment—numerical features—and the results clearly demonstrate that this is the case. This outcome is expected, since from a theoretical perspective, the projection used to split the examples in each tree node based on the Euclidean distance in a single dimension (on a single feature) should produce the same order of examples as the original feature. This means, that the differences between Random Forests and Random Similarity Forests are probably due to some implementation details and some random variability. However, the significantly worse performance of Similarity Forests on two datasets showcases that it should be used with a little more caution, which confirms the recent findings of Czekalski and Morzy [17].

E. Evaluation on Complex Data

The second experiment was designed to assess the performance of Random Similarity Forests on complex data. This type of data should be a natural environment for Similarity Forests, so analogously to the first experiment, the aim of this comparison is to verify if RSF can match SF on this kind of data. Importantly, this is also the first reported test of Similarity Forests on actual complex data without explicitly defined features, since the algorithm was only tested on scalar data in the original paper [5] and the follow-up study [17].

In this group of experiments, both SF and RSF rely on the same distance measure for projection, namely, the edit distance. Edit distance calculates the minimum total cost of operations required to transform one sequence into another using three single-element operations: insertion, deletion, and relabeling. Since we are not dealing with regular sequences but, more complex, sequences of sets (i.e., sequences where each element is a set), the measure is slightly adapted to this type of data by using the Jaccard distance as the cost function for element relabeling. Given two sets s_i and s_j the Jaccard distance is defined as $\delta(s_i, s_j) = 1 - \frac{|s_i \bigcap s_j|}{|s_i \bigcup s_j|}$.

For completeness, we also wanted to include RF in this comparison, however, they are incapable of processing complex data in their original form. Therefore, for this classifier, we perform an additional feature extraction step, in which the datasets are decomposed into simple, numerical features. More precisely, we rely on a bag of words decomposition, where each sequence of sets is represented as a union of all sets in a given sequence. Formally, given a sequence of sets S, it can be expressed as $bow(S) = \bigcup_{i=1}^{|S|} s_i$, where s_i is the i^{th} set in a given sequence. After such a decomposition is complete, we can encode the obtained data as a feature vector using one-hot encoding. The results of this experiment are presented in Table I and Fig. 5.

Since the complex datasets were designed to include a varied degree of different types of information, we will discuss the results one dataset at a time in the order of increasing sequential information. The dataset in which the examples in the two classes are differentiated only by the set information and not by sequential information is `items`. Here, we can observe that a simple bag of words representation with binary encoding is able to easily distinguish between the two classes, with RF significantly outperforming SF and RSF. However,

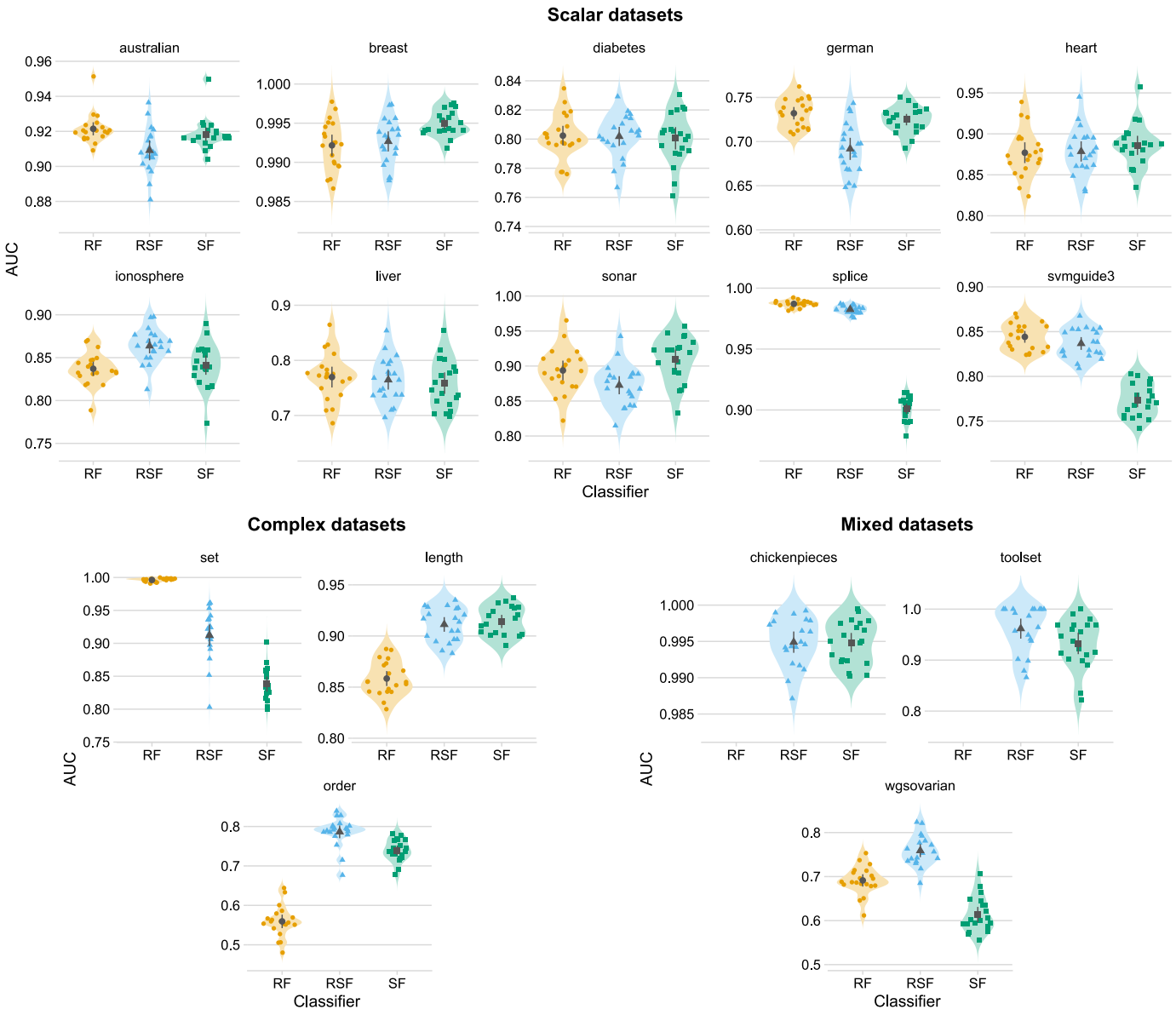

Fig. 5: Comparative evaluation of Random Forests (RF), Similarity Forests (SF), and Random Similarity Forests (RSF) on datasets with scalar features (top), complex features (bottom left) and mixed types of features (bottom right). Black points and error bars show the mean and 95% confidence interval of each 10 $\times$ 2-fold cross-validation experiment.

it is important to remember that it did not actually operate on the original data and required an additional preprocessing step. Nevertheless, this result clearly demonstrates that for easy problems, relying on approaches based on complex distance measures may actually be detrimental to the quality of predictions. In this case, the sequential information only obfuscated the true source of class information and, hence, the decision boundary. This outcome is actually expected, since the dataset is constructed so as the items in each consecutive set are selected independently from the previous sets, which means that this dataset represents sets of sets rather than sequences of sets.

Another interesting observation from these results is that RSF significantly outperformed SF. This is an unexpected result since both approaches rely on the same mechanism and the same projection. Because the dataset does not have any features, RSF simply treats all the objects as a single feature, which is exactly how SF processes examples. The only identifiable difference is in the details of the split selection step. Both approaches select the best split according to the Gini index, however, in the case when several splits achieve the same score, RSF tries to split the examples as evenly as possible, while in the case of SF the issue of ties is unaddressed. On the other hand, this difference in performance

may be also due to some other differences in implementation details.

The second dataset in the order of sequential information content is `length`. Here, examples in the two classes are differentiated only by the lengths of the sequences. Given this fact, the results of this experiment are somewhat surprising. First of all, unsurprisingly, RSF and SF both significantly outperformed RF. This is expected as the information about the length of the sequences is very easy to capture through edit distance as each additional element in one of the sequences extends the distance by 1. From that perspective, the binary-encoded bag of words representation used by RF should not be able to detect this information, which is why its high performance is to some extent surprising, especially considering the fact that the items in the sets of the sequences in both classes are drawn from exactly the same distribution. However, since the sequences in one class are longer than in the other one, more unique items have a chance to fall into the sets in this class. This leads to a denser representation in these examples which is clearly enough to distinguish between the two classes with relatively high accuracy.

Comparing RSF to SF, we can conclude that there is no difference in performance between the two approaches on this dataset. As with all other datasets in this category, this is an expected outcome which once again confirms that the proposed algorithm is capable of dealing with complex data on par with Similarity Forests.

The third and final dataset in the complex data category is `order`. Here, the difference between the two classes is even more difficult to identify as the elements in one of the classes are ordered while in the other they are not. Since the ordering is stochastic (i.e., the items in each consecutive set are drawn from a slightly right-shifted distribution), the overall statistics of the sequences in both classes are nearly identical. This is clearly reflected in the results as RF was barely able to find any regularities in the data through the lens of binary-encoded bag of words and was significantly outperformed by both, SF and RSF. Comparing the two latter approaches, we can observe that RSF also significantly outperformed SF. As already discussed with the `items` dataset, this is somewhat unexpected and can be due to the difference in how the algorithms deal with choosing between equally good splits or some other implementation details.

F. Evaluation on Mixed Data

The final experiment was designed to evaluate Random Similarity Forests on datasets with a mix of simple, numerical features and/or complex, object-like features. These are the type of datasets that neither Random Forest nor Similarity Forest can process out of the box. Since two of the datasets (`chickepieces`, `toolset`) in this experiment are defined by pre-computed distances between each pair of examples, they cannot be used by RF. Furthermore, to be able to evaluate SF on the whole variety of available pre-computed distances, we run the algorithm on all distances separately and take the median score. In order to compare the final dataset (`wgsovarian`) on all three algorithms, we preprocess this dataset for RF and SF, analogously to the experiment with complex data. In this case, in the preprocessing step each of the six features representing the histograms of the lengths of various structural variants is encoded simply as the total number of such variants. The remaining 38 numerical features remain unchanged. This way, after transformation the resulting dataset has the same number of features as the original one. For distance measures in this dataset, in RSF we rely on the Euclidean distance for both numerical and histogram features as suggested by the research on measuring distances between histograms [28]. Similarly, SF also use the Euclidean distance to calculate the distances over the whole 44 dimensional feature space. The results of this experiment are presented in Table I and Fig. 5.

The results show that RSF outperforms SF on the two datasets with pre-processed distances and outperforms both RF and SF on the `wgsovarian` dataset. This clearly demonstrates that the additional information accessible through the mixture of complex features is important from the classification perspective as RSF is the only algorithm capable of fully incorporating it. It is further worth noting that SF is also outperformed by RF on the `wgsovarian` dataset, even though they operate on exactly the same data. This can be explained by the fact that the 6 complex features are of higher importance than the remaining 38 numerical features. Therefore, when SF processes entire examples using a single distance measure, their effect is dampened by the remaining, more numerous features.

G. Summary of the Results

The main takeaways from the experiments are as follows.

- The proposed algorithm was able to match the performance of Random Forests and match or outperform Similarity Forests on datasets consisting of scalar features.
- Random Similarity Forests were also able to match the performance of Similarity Forests on complex data.
- Random Forests combined with scalar feature extraction may be more appropriate than Random Similarity Forests or Similarity Forests with complex distance measures for complex data types with easy to identify decision boundaries.
- Random Similarity Forest was the only algorithm capable of using a mixture of scalar and complex features. This property helped Random Similarity Forests outperform Random Forests and Similarity Forests on datasets with both types of features. The experiments have shown that for real-world genomic data transforming complex features to scalars or using a single example-wide distance measure may result in information loss.
- Due to the increased randomness, Random Similarity Forests usually require larger feature samples (*max_features*) than are commonly used for Random Forests.

V. Conclusions

In this paper, we have addressed the problem of classifying data with a mixture of scalar and complex features by proposing a new classifier, called Random Similarity Forest. Like other decision forest algorithms, such as Random Forests and Similarity Forests, the proposed algorithm relies on bagging of decision trees, but in contrast to other approaches splits the tree nodes according to distance-based projections of single feature values. This allows for a very flexible classification approach, in which traditional numerical features can be used alongside complex object-like features with domain-specific distance measures, taking full advantage of each data type. The proposed method was experimentally evaluated against Random Forests and Similarity Forests on datasets with scalar, complex, and mixed data features, showcasing its capabilities across all possible data domain configurations.

Experimental results on 16 datasets show that Random Similarity Forests can be considered a safe alternative to the well-established Random Forests on scalar datasets and are on par or exceed the performance of Similarity Forests on complex data. Most importantly, however, the experiments show that the proposed classifier was the only one inherently capable of dealing with datasets consisting of a mixture of scalar and complex features, and outperformed both of the alternatives on datasets with mixed data types.

The method proposed in this paper opens many new avenues for further research. Regarding the classifier itself, it would be interesting to verify the impact of using multiple distance measures for each feature. Random Similarity Forests could also be extended to use multi-feature distance measures at split nodes. Moreover, the effects of other forms of randomization on the proposed algorithm, such as random splits in the projection space, could also be inspected. Finally, it will be interesting to observe what types of features and distance measures will prove useful when Random Similarity Forests are applied to real-world data from various domains.

Acknowledgment

This publication and the underlying study have been made possible by the data that the Hartwig Medical Foundation and the Center of Personalized Cancer Treatment (CPCT) have made available. This work was also supported by data obtained from ICGC Breast Cancer Working Group and The Cancer Genome Atlas.

References

[1] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *J. Mach. Learn. Res.*, vol. 15, no. 1, p. 3133–3181, 2014.

[2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nat.*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] E. Fix and J. Hodges, "Discriminatory analysis, nonparametric discrimination: Consistency properties usaf school of aviation medicine, randolph field," Texas, Tech. Report 4, Tech. Rep., 1951.

[5] S. Sathe and C. C. Aggarwal, "Similarity forests," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, p. 395–403.

[6] H. Davies et al., "HRDetect is a predictor of BRCA1 and BRCA2 deficiency based on mutational signatures," *Nature Medicine*, vol. 23, no. 4, pp. 517–525, 2017.

[7] G. E. Hinton and M. Revow, "Using pairs of data-points to define splits for decision trees," in *Advances in Neural Information Processing Systems*, D. Touretzky, M. C. Mozer, and M. Hasselmo, Eds., vol. 8. MIT Press, 1996.

[8] E. Scornet, "Random forests and kernel methods," *IEEE Transactions on Information Theory*, vol. 62, no. 3, pp. 1485–1500, 2016.

[9] K. Shestopaloff, M. Dong, F. Gao, and W. Xu, "Dcmd: Distance-based classification using mixture distributions on microbiome data," *PLOS Computational Biology*, vol. 17, no. 3, p. e1008799, 2021.

[10] T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka, "Distance-Based Image Classification: Generalizing to new classes at near-zero cost," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2624–2637, 2013.

[11] M. Neuhaus and H. Bunke, "Edit distance-based kernel functions for structural pattern classification," *Pattern Recognition*, vol. 39, no. 10, pp. 1852–1863, 2006.

[12] A. Abanda, U. Mori, and J. A. Lozano, "A review on distance based time series classification," *Data Mining and Knowledge Discovery*, vol. 33, no. 2, pp. 378–412, Mar. 2019.

[13] C.-F. Tsai, W.-Y. Lin, Z.-F. Hong, and C.-Y. Hsieh, "Distance-based features in pattern classification," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, p. 62, 2011.

[14] M. Piernik and T. Morzy, "A study on using data clustering for feature extraction to improve the quality of classification," *Knowledge and Information Systems*, 2021.

[15] J. Liang, Q. Liu, N. Nie, B. Zeng, and Z. Zhang, "An improved algorithm based on knn and random forest," in *Proceedings of the 3rd International Conference on Computer Science and Application Engineering*, 2019.

[16] P. S. Reel, S. Reel, E. Pearson, E. Trucco, and E. Jefferson, "Using machine learning approaches for multi-omics data analysis: A review," *Biotechnology Advances*, vol. 49, p. 107739, 2021.

[17] S. Czekalski and M. Morzy, "Similarity forests revisited: A swiss army knife for machine learning," in *Advances in Knowledge Discovery and Data Mining*, 2021, pp. 42–53.

[18] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, p. 3–42, 2006.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Wadsworth International Group, 1984.

[20] J. Huang and C. X. Ling, "Using AUC and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[21] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[24] L. Breiman, "Arcing classifiers," *Annals of Statistics*, vol. 26, pp. 801–823, 1998.

[25] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[26] A. R. Benson, R. Kumar, and A. Tomkins, "Sequences of sets," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, p. 1148–1157.

[27] B. Spillmann, "Description of the distance matrices," 2014. [Online]. Available: https://fki.tic.heia-fr.ch/databases/string-edit-distance-matrices

[28] S.-H. Cha and S. N. Srihari, "On measuring the distance between histograms," *Pattern Recognition*, vol. 35, no. 6, pp. 1355–1370, 2002.