Interpretable Instance-Based Learning through Pairwise Distance Trees

Andrea Fedele[1,2] ✉, Alessio Cascione[1], Riccardo Guidotti[1,2], and Cristiano Landi[1,2]

[1] University of Pisa, Largo Bruno Pontecorvo 3, Pisa PI 56127, Italy
{name.surname}@phd.unipi.it, {riccardo.guidotti}@unipi.it
[2] KDD Lab, ISTI-CNR, Via G. Moruzzi 1, Pisa PI 56124, Italy
{name.surname}@isti.cnr.it

Abstract. Instance-based models offer natural interpretability by making decisions based on concrete examples. However, their transparency is often hindered by the use of complex similarity measures, which are difficult to interpret, especially in high-dimensional datasets. To address this issue, this paper presents a meta-learning framework that enhances the interpretability of instance-based models by replacing traditional, complex pairwise distance functions with interpretable pairwise distance trees. These trees are designed to prioritize simplicity and transparency while preserving the model's effectiveness. By offering a clear decision-making process, the framework makes the instance selection more understandable. Also, the framework mitigates the computational burden of instance-based models, which typically require calculating all pairwise distances. Leveraging the generalization capabilities of pairwise distance trees and employing sampling strategies to select representative subsets, the method significantly reduces computational complexity. Our experiments demonstrate that the proposed approach improves computational efficiency with only a modest trade-off in accuracy while substantially enhancing the interpretability of the learned distance measure.

Keywords: Pairwise Learning · Interpretable Distance · Meta-learning.

1 Introduction

Instance-based models, such as k-Nearest Neighbors, have long been valued for their intuitive approach: they make predictions by comparing new instances to stored examples from training data [1, 32]. This case-based reasoning, rooted in how humans naturally use past experiences to understand new situations, offers inherent interpretability by relying on concrete examples [31]. Instance-based models rely on pairwise comparison: each new instance is evaluated against stored examples using a distance measure. However, while the decision mechanism is typically transparent, e.g., majority voting over nearest neighbors, the process by which these neighbors are determined is tied to the underlying distance measure, which often remains opaque, especially in high-dimensional spaces where is complex to determine the contribution of single features.

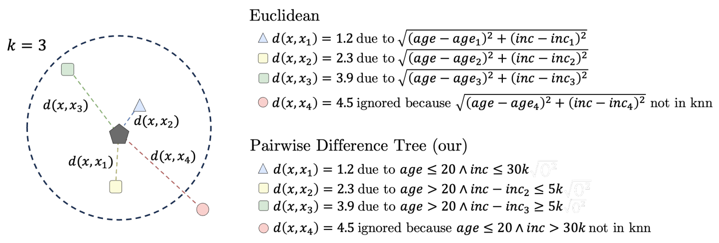
$k = 3$
$d(x, x_3)$
$d(x, x_2)$
$d(x, x_1)$
$d(x, x_4)$
Euclidean
△ $d(x, x_1) = 1.2$ due to $\sqrt{(age - age_1)^2 + (inc - inc_1)^2}$
□ $d(x, x_2) = 2.3$ due to $\sqrt{(age - age_2)^2 + (inc - inc_2)^2}$
□ $d(x, x_3) = 3.9$ due to $\sqrt{(age - age_3)^2 + (inc - inc_3)^2}$
○ $d(x, x_4) = 4.5$ ignored because $\sqrt{(age - age_4)^2 + (inc - inc_4)^2}$ not in knn
Pairwise Difference Tree (our)
△ $d(x, x_1) = 1.2$ due to $age \leq 20 \land inc \leq 30k$
□ $d(x, x_2) = 2.3$ due to $age > 20 \land inc - inc_2 \leq 5k$
□ $d(x, x_3) = 3.9$ due to $age > 20 \land inc - inc_3 \geq 5k$
○ $d(x, x_4) = 4.5$ ignored because $age \leq 20 \land inc > 30k$ not in knn

Fig. 1: Comparison between traditional KNN using Euclidean distance, and KNN using PDT distance, i.e., PDTF with KNN at inference time.

In contrast to traditional instance-based methods, Pairwise Difference Learning (PDL) and Pairwise Similarity Learning (PSL), have emerged as techniques that leverage pairwise relationships by operating directly on input pairs. PDL shifts the focus from mapping individual inputs directly to outputs toward learning (regression) functions that predict the difference between outcomes for pairs of instances [35, 38]. The core idea is to approximate the difference function, enabling predictions for a new test instance by averaging the predicted differences relative to the outcomes of the training data. In contrast, PSL emphasizes learning similarity functions that assign higher scores to pairs of samples from the same class than to those from different classes [13, 25]. Both methods are particularly valuable in data-scarce scenarios, such as rare disease diagnosis or novel phenomenon detection, where traditional data-intensive approaches may falter. However, both frameworks have limited interpretability, as they generally depend on complex or high-dimensional transformations.

Motivated by the need for both robust pairwise modeling and interpretable instance-based decision-making, we introduce the Pairwise Distance Tree Framework (PDTF), an interpretable meta-learning approach that enhances instance-based models by replacing stpdtandard, opaque distance functions with a shallow, interpretable decision tree, namely the Pairwise Distance Tree (PDT). By operating on pairwise representations of the input data, PDTF unifies the strengths of PDL and PSL while providing a clear explanation of why two instances are considered (dis)similar. The PDT learns a mapping from a joint representation of instances to their corresponding distance, with its decision rules explicitly revealing which features drive the similarity judgment. To further boost interpretability, we also introduce forced split conditions, ensuring that the same features and the same thresholds are used when comparing both elements of a pair. In Figure 1, we compare the decision-making process of a "traditional" KNN using Euclidean distance with that of KNN using the PDT approximated distance. Given the same neighborhood with $k = 3$, Euclidean distance requires mathematical calculus that hardly explain why the blue, yellow and green points are included while the red point is excluded, especially in high-dimensional spaces. In contrast, PDTF offers an explanation in a logical form for neighborhood selection that users can easily understand without intricate mathematical reasoning.

Our contribution is threefold. First, we model the distance function as an interpretable decision tree, offering full visibility into the instance-based reasoning process. Second, we integrate pairwise difference and similarity-based instance selection within a unified framework that emphasizes transparency. Third, we address the computational challenges inherent in pairwise methods through efficient instance sampling strategies. Experiments on tabular benchmark datasets reveal that a joint feature representation based solely on pointwise differences yields the best performance for PDT distance approximation, while the combined representation of pointwise differences and input features excels in the classification tasks. Moreover, an intelligent sampling strategy, using roughly 20% of the dataset, reduces training time without significantly compromising performance. Also, enforcing split constraints enhances interpretability without compromising performance by reducing the cognitive burden of interpreting rules. Since the splits are restricted to the same feature, or both the same feature and threshold, the resulting rules is simpler to understand.

The rest of the paper is organized as follows. In Section 2 we review related work covering case-based reasoning and similarity learning. Section 3 describes the details of the PDTF, including the pairwise training set preparation, tree structure, and forced split constraints. Section 4 presents the experimental results, and Section 5 concludes with a discussion of future research directions.

2 Related Work

Case-based reasoning relies on the idea that human cognition often uses stored examples of past experiences to interpret new situations [33]. Decisions are made by retrieving and comparing instances from memory, an inherently interpretable process, as users can trace decisions to concrete examples [22]. Its transparency has driven applications in healthcare [3], financial risk [24], and image analysis [18], where understanding decision rationale is as vital as the decision itself.

Building on these insights, metric learning, deep metric learning and Pairwise Similarity Learning (PSL) methods have been developed to capture the intrinsic relationships between data points. Metric learning seeks to learn a proper distance function that satisfies non-negativity and triangle inequality properties and respects semantic simi larity by reducing intra-class distances and enlarging inter-class distances [39]. However, in high-dimensional settings, such learned metrics can become less interpretable as the individual contributions of features are obscured by the complexity of the transformation. Deep metric learning extends these ideas using neural networks to learn non-linear embeddings that have achieved state-of-the-art performance in tasks such as face recognition and image retrieval [21, 23, 40]. However, despite their success, the non-linear transformation involved lead to opaque learned metrics. In contrast, PSL focuses on learning a similarity function that assigns higher scores to positive pairs than to negative pairs, thereby dropping the requirement of learning a proper distance metric. PSL methods are categorized into proxy-based and proxy-free approaches [37]. Proxy-based methods use representative vectors (proxies) for each class to compute similarity, improving convergence but reducing transparency by adding an abstraction layer. Proxy-free methods, in contrast, work directly on data pairs or triplets, providing a more intuitive and transparent view of the data's structure. Nevertheless, even these approaches struggle with interpretability in high-dimensional settings, where hyperparameter sensitivity (e.g., in triplet loss formulations) and the complexity of learned representations can obscure the underlying feature contributions.

A fundamental contribution towards this line of research is the Pairwise Difference Learning (PDL) [35]. Rather than mapping individual inputs directly to outputs, the PDL framework shifts the paradigm by learning a function that predicts as a regressor the difference between outcomes for pairs of instances. Predictions are obtained for a given test instance by pairing it with all training examples, computing the corresponding outcome differences, and averaging these values. This meta-learning approach is applied across various fields, including image processing [20], drug activity ranking [36], and quantum mechanical reaction modeling [9]. The first extension of PDL to classification is introduced in [2] where the classification problem is reformulated as a binary task: a paired dataset is constructed, and a binary classifier is trained on joint feature representations to predict whether a pair of instances belongs to the same class. For new test samples, pairwise predictions are aggregated to estimate class posterior probabilities, harnessing the robustness and natural uncertainty quantification of pairwise comparisons. Although relatively recent, there is emerging interest in combining ideas from PSL and PDL. In [11], for example, the authors address the authorship analysis problem by representing a feature vector as an unordered pair of documents. Here, the value of a feature is computed as the absolute difference in the relative frequencies of that feature across the two documents. Similar to the formulation in [2], the class label indicates whether the two documents belong to the same author.

Traditional case-based reasoning and modern pairwise learning methods enhance predictive performance but lack transparency in their distance functions. To address this, we unify the strengths of PSL and PDL with a focus on interpretability. We reformulate the pairwise learning task as a regression problem, mapping input pairs to distance values. This distance function is modeled with a shallow regression tree, which approximates the original distance while providing explicit decision rules which offer clear decision paths, improving transparency and enabling human understanding of how distances are determined.

3 Pairwise Distance Tree Framework

We present here the Pairwise Distance Tree Framework (PDTF), an interpretable meta-learning framework designed to improve the transparency of instance-based models. PDTF first transforms training instances into pairs with computed distances, then trains a shallow regressor tree to approximate these distances. Finally, the learned function replaces the standard distance to guide neighbor selection. In the following, we first formalize the problem setting and then we present our framework. Without loss of generality, we focus on classification tasks, leaving the exploration of other problem domains to future work.

3.1 Problem Setting

Given a set of instances represented as real-valued m-dimensional feature vectors[3] in $\mathbb{R}^m$ and a set of class labels $C = \{1, \dots, c\}$, we assume the existence of an unknown ground-truth function $g : \mathbb{R}^m \to C$ mapping each vector in $\mathbb{R}^m$ to one of the c classes in C. In case-based reasoning, given a training set $\langle X, Y\rangle$ with $X = \{x_1, \dots, x_n\}$ of n instances, $Y = \{y_1, \dots, y_n\}$ denoting the corresponding class labels with $y_i \in C$, and a pairwise distance function $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$, the objective is to learn an instance-based model implemented through function $f : \mathbb{R}^m \to C$, which aims to approximate the unknown ground-truth function g. Instance-based models explicitly store a set of the training data $\langle X, Y\rangle$, referred to as the *memory*, which is used during inference. These models define a selection policy s based on d such that, at inference time, given a memory $\langle X, Y\rangle$ and a query instance x, the selection policy is applied to identify a subset of cases

$$\langle X_s, Y_s\rangle = s_d(x, \langle X, Y\rangle) \quad \text{where } X_s \subseteq X \text{ and } Y_s \subseteq Y.$$

This subset typically consists of the closest examples, commonly referred to as *neighbors* [16, 34] or *pivots* [4, 8], which are then used to make a prediction. In essence, the selection policy is a similarity-based mechanism that identifies a set of k neighbors of the query instance x. Once the relevant instances are selected, an inference policy ϕ is applied, usually a majority vote over the class labels of the selected instances $\langle X_s, Y_s\rangle$. Thus, the instance-based model f is defined as:

$$f(x) = \phi(\{(x_i, y_i) \mid (x_i, y_i) \in s_d(x, \langle X, Y\rangle)\})$$

Common hyper-parameters of the selection policy s include a similarity threshold, which determines whether an instance is selected w.r.t. the distances calculated, or a value k representing the number of most similar instances to retrieve.

Typical instance-based models include k-Nearest Neighbor (KNN) [16,34], and more broadly, most case-based reasoners [30]. They are considered interpretable in terms of similarity, as they rely on a set of previously observed "cases" that serve as evidence during inference. This inherent interpretability holds as long as the following conditions are met: *(i)* the inference policy ϕ uses the selected evidence in an interpretable manner, *(ii)* the selection policy s transparently identifies relevant instances from memory, and *(iii)* the pairwise distance function d is interpretable. Regarding *(i)* and *(ii)*, most instance-based models, such as those applying a majority vote and selecting the k most similar instances to a query instance x, are generally considered interpretable. However, full interpretability hinges on the notion of similarity itself, which depends on the distance

[3] For the sake of simplicity, we consistently treat data instances as real-valued vectors. Any transformation employed in the experiments is specified when needed.

function d. Specifically, an instance-based model can be deemed entirely human-interpretable only if the distance function: *(a)* is transparent, and *(b)* relies on a limited number of features to compare instances and establish similarity. For example, consider a classification task on a tabular dataset with $m = 100$ features, where the KNN model uses the Euclidean distance as d. While the decision process over the neighborhood of k instances is interpretable, a user may struggle to fully understand why certain instances are included in the neighborhood while others are not, due to the high dimensionality, without replicating the mathematical calculations to compute the distance.

3.2 Pairwise Distance Tree

To overcome the aforementioned limitations, we implement the distance function d with a shallow Pairwise Distance Tree (PDT) that allows to express the reasons why two instances are similar or dissimilar only considering a limited number of features and expressing the reasons for the distance in a logical form.

We opt for implementing the distance function d with a tree-based model because decision trees are interpretable predictive models [17,19] representing their decisions through a structure composed of nodes and branches [7,34]. Indeed, a decision tree routes instances within their structure, each node testing a split condition on feature a w.r.t. threshold τ, e.g., $x^{(a)} \leq \tau$, and routing instances towards its children, all the way down to the leaf nodes. Thus, decision trees are inherently transparent because the complete tree can be inspected, allowing a human analyst to follow the sequence of splits. Each instance traces a path inside the tree, effectively providing a *decision rule* describing the decision process of the tree on the said instance. The *complexity* [28] of decision trees is typically calculated as the total number of nodes and leaves, tree depth, and number of attributes used. The simpler the tree, the more concise and interpretable the decision rules [10,14,15]. Tree induction algorithms typically implement a top-down greedy search through the space of possible splits. CART [7], ID3 [26], and its successor C4.5 [27] are the most famous induction strategies.

Pairwise Training Set Preparation. Given a training set $\langle X, Y\rangle$ and a distance function d, we transform $\langle X, Y\rangle$ into a paired dataset $\langle Z, D\rangle$ where

$$Z = \{z_{ij} = \zeta(x_i, x_j) \mid x_i, x_j \in X\} \quad D = \{d_{ij} = d(x_i, x_j) \mid (x_i, x_j) \in Z\}$$

where z_{ij} is formed by using the feature vectors x_i and x_j. In particular, we consider the three following alternatives to implement the transformation ζ:

- (α) the *concatenation* of the feature vectors $z_{ij} = \zeta_\alpha(x_i, x_j) = [x_i, x_j]$,
- (β) the *pointwise difference* of the feature vectors $z_{ij} = \zeta_\beta(x_i, x_j) = |x_i - x_j|$ that is a formulation shown to positively impact performance [35], and
- (γ) the combination of α and β, i.e., $z_{ij} = \zeta_\gamma(x_i, x_j) = [x_i, x_j, |x_i - x_j|]$.

We underline that, since we want to reflect symmetry in our framework, we add both (x_i, x_j) and (x_j, x_i) to the paired dataset. Also, in order to take into account cases in which the distance is zero ($d_{ij} = 0$), we consider in Z also cases in which $i = j$, i.e., (x_i, x_i). Thus, the maximum number of pairs in Z, i.e., the maximum cardinality of $|Z|$ is n^2 where $|X| = n$. Therefore, our proposal is named Pairwise Distance Tree, as it takes as input the domain $\langle Z, D\rangle$ formed by pairing the original training iances and using their distance as target variable.

Also, to address the complexity associated with pair creation, we present a set of *sampling strategies* to consider a subset of training instances $\tilde{X} \subset X$ with $|\tilde{X}| = \tilde{n} < n$ in order to reduce the number of pairs in Z:

- *random (RS):* selects uniformly at random $\tilde{n}$ instances among those in $\langle X, Y\rangle$;
- *center-based clustering (CS):* executes a k-Means clustering algorithm for each target label among those in $\langle X, Y\rangle$ by setting $k = \lceil \tilde{n}/c \rceil$ and selects for each cluster the closest instance to the centroid.

Pairwise Tree Structure. The Pairwise Distance Tree is implemented as a decision tree regressor r that maps the joint representation z_{ij} to an approximation of the distance $d(x_i, x_j)$, i.e., $d(x_i, x_j) \approx r(x_i, x_j)$. The goal is to learn a decision tree regressor r such that the prediction of $r(x_i, x_j)$ closely match the true distance $d(x_i, x_j)$. This is achieved by minimizing regression loss over all pairs. Therefore in PDT, each decision path provides a clear, step-by-step explanation of how the model evaluates pairwise distances.

In this context the structure of the tree regressor r is crucial. By designing r as a shallow decision tree, we ensure that its decision-making process is interpretable, thereby allowing us to inspect and understand the logic adopted to approximate the distance between a pair of instances. We underline that, adopting a shallow regression tree, each leaf is returning as distance the *average* distance among a consistent group of similar pairs that the training procedure routed in that leaf. Thus, if a PDT has l leaves, e.g. $l = 16$ leaves, it means that, at inference time only l values can be returned by $r(x_i, x_j)$. This behavior, on the one hand, is a strong limitation w.r.t. the calculus of traditional distance functions because the approximation applied by the PDT practically applies a discretization to the original pairwise distances, on the other hand, is well-aligned with the human way of estimating the similarity between objects as we simply say they are *very different*, they are *different*, *similar* or *very similar*.

Split Constraints. To further boost the interpretability of PDT, when the feature selected for the best split belongs to the feature vectors (x_i, x_j) in the training set preparation α and γ, we impose two *forced split conditions* which we refer to as the *same-feature*, and the *same-feature, same-threshold* splits.

Under the *same-feature* split (PDT-F), if a parent node (which is not itself forced) splits on a specific feature a from one element of the pair x_i, i.e., $x_i^{(a)} \leq \tau$, then, the corresponding children nodes are forced to split on the same feature a from the other element of the pair x_j, without any constraint on the threshold, i.e., $x_j^{(a)} \leq \tau'$, allowing the tree only to select the threshold τ'. This constraint ensures that both components of the pair are evaluated along the same dimension. The key point is that the decision process explicitly links the two splits by enforcing the use of the second feature on both records. On the other hand, the *same-feature, same-threshold* split (PDT-T) requires that the children nodes

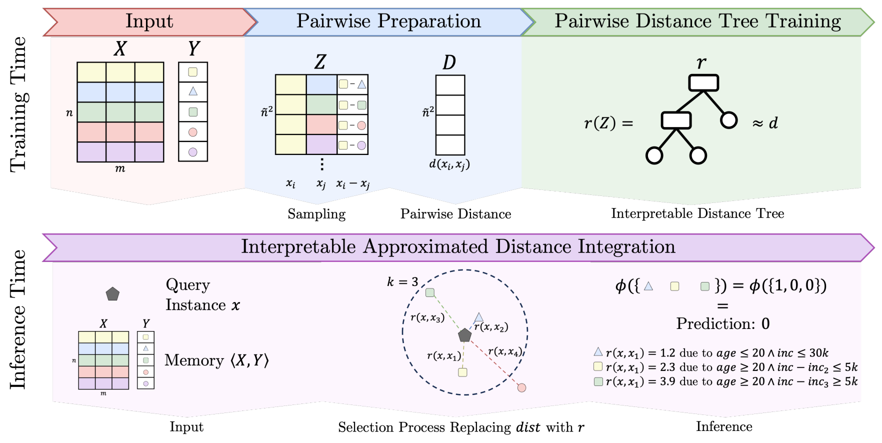
Input
Pairwise Preparation
Pairwise Distance Tree Training
Training Time
X
Y
n
m
Z
$\tilde{n}^2$
x_i
x_j
$x_i - x_j$
D
$d(x_i, x_j)$
r
$r(Z) =$
$\approx d$
Sampling
Pairwise Distance
Interpretable Distance Tree
Interpretable Approximated Distance Integration
Inference Time
Query Instance x
Memory $\langle X, Y \rangle$
$k = 3$
$r(x, x_3)$
$r(x, x_2)$
$r(x, x_1)$
$r(x, x_4)$
$\phi(\{ \triangle\ \square\ \square \}) = \phi(\{1, 0, 0\})$
=
Prediction: 0
$r(x, x_1) = 1.2$ due to $age \leq 20 \wedge inc \leq 30k$
$r(x, x_1) = 2.3$ due to $age \geq 20 \wedge inc - inc_2 \leq 5k$
$r(x, x_1) = 3.9$ due to $age \geq 20 \wedge inc - inc_3 \geq 5k$
Input
Selection Process Replacing $dist$ with r
Inference

Fig. 2: Pairwise Distance Tree Framework (PDTF). At inference time, given a query instance x, the model selects relevant neighbors from the memory $\langle X, Y \rangle$ by evaluating $r(x, x_i)$ and applies an inference policy ϕ, i.e., majority voting, to produce a final prediction. Each prediction can be inspected, since the distance function employed for neighborhood selection is fully interpretable.

not only uses the same feature a as the parent, but also applies the exact same threshold τ, thereby creating a strict alignment between the decision paths for both elements. Thus, if the parent node perform the split $x_i^{(a)} \leq \tau$, the children nodes are constrained to perform splits $x_j^{(a)} \leq \tau$. This constraint further enhances the consistency in the splits across two elements of each pair. The intuition behind these forced split conditions w.r.t. the plain PDT (PDT-P) is that interpretability is enhanced when the decision process consistently evaluates the same features, and, in the strictest case, uses identical thresholds across both elements of a pair. When the splits are aligned, a human reviewer can clearly identify which features are driving the decision and understand how differences between the records are being measured.

3.3 Pairwise Distance Tree Framework

Given a training set $\langle X, Y \rangle$ and a distance function d, the PDTF learns an instance-based model f by adopting a PDT regressor r to approximate d. In summary, PDTF consists of three main steps illustrated also in Figure 2:

1. *Pairwise Training Set Preparation:* transform the dataset $\langle X, Y \rangle$ (ora a subset of it) into the paired dataset $\langle Z, D \rangle$ using the joint representation z_{ij} and the distance d_{ij} of a sample pair (x_i, x_j), in blue in Figure 2.
2. *Pairwise Distance Tree Training:* train PDT regressor r by minimizing a regression loss over all the selected pairs, thereby learning an interpretable mapping from z_{ij} to d_{ij}, in green in Figure 2.

3. *Interpretable Approximated Distance Integration:* replace d in the instance-based model f with the interpretable approximated distance function r to select the cases to take the decision w.r.t. a query instance x, i.e., $\langle X_s, Y_s \rangle = s_r(x, \langle X, Y \rangle)$ where for a query instance x and any training sample x_i, the approximated distance is computed as $r(x, x_i)$, in purple in Figure 2.

This decoupled structure is critical because the regression training phase leverages only a subset $\tilde{X} \subset X$ of input pairs to learn the distance function, while the full training set $\langle X, Y \rangle$ is retained as the input for the instance-based model f. This design allows us to efficiently learn a robust, interpretable distance function without sacrificing the comprehensive information provided by the complete dataset during inference. However, we signal to the reader that the learned nature of our interpretable distance function does not theoretically guarantee that all traditional metric properties, such as *symmetry*, are strictly met. However, by including both orderings of instance pairs (x_i, x_j) and (x_j, x_i), whether $i = j$ or $i \neq j$, in the paired training set, we empirically achieve *approximated symmetry*[4]. Thus, at inference time, the computed distance for a query instance x and any training sample x_i is robust to the ordering of inputs, meaning that $r(x, x_i) \approx r(x_i, x)$.

In addition, we underline that, compared to the PDL classifier [2], our proposal is able to capture a much finer and detailed abstraction of the notion of distance between pair of instance while simultaneously sufficiently abstracting from the original distance function. Furthermore, due to the sampling strategies used to construct the pairwise distance tree training set, PDTF is computationally more efficient than the PDL classifier. Given $n = |X|$, the training complexity of PDTF, like that of PDL, is primarily determined by the calculation of the pairwise distance matrix, which requires $O(n^2)$ operations. Indeed, the complexity of training the PDT itself is $O(m \cdot n \log^2 n)$ for balanced trees. When employing a sampling strategy, the complexity of PDTF depends on the sampling strategy and on the reduced dataset size $\tilde{n}$, where $\tilde{n} < n$, further improving computational efficiency. In particular, when using random sampling strategies, the overall training complexity is $O(\tilde{n}^2)$, omitting the dataset dimensionality m for simplicity. On the other hand, when adopting the center-based clustering sampling strategy, the complexity depends on the clustering algorithm employed. If, as in our case, k-Means is used with $k = \lceil \tilde{n}/c \rceil$, and considering that the complexity of k-Means can be approximated as $O(k \cdot n)$, the overall training complexity becomes $O(\tilde{n} \cdot n)$. At prediction time, the complexity of PDTF for a single query instance is $O(n \cdot \log \tilde{n})$, which is more efficient than the $O(n \cdot m)$ complexity of the traditional KNN classifier or the PDL classifier when $\log \tilde{n} < m$.

[4] Approximated symmetry means that training PDT on both orderings of instance pairs (x_i, x_j) and (x_j, x_i), achieves prediction vectors with very high cosine similarity.

Table 1: Datasets info: # of records (n), # of features (m), # of classes (c).

	small									large										
	`iris`	`seeds`	`glass`	`fire`	`verteb.`	`ecoli`	`iono`	`lrs`	`breast`	`steel`	`road`	`bank`	`pol`	`cover`	`house`	`eye`	`sylvine`	`magic`	`compas`	`spam`
n	150	210	214	243	310	336	351	531	569	1.9k	2k	2k	2k	2k	2k	2k	2k	2k	4.5k	4.6k
m	4	7	9	13	6	7	34	100	30	27	29	7	26	10	16	20	20	20	9	57
c	3	3	6	2	2	8	2	9	2	7	2	2	2	2	2	2	2	10	2	2

4 Experiments

We evaluate the performance of PDTF[5], on tabular benchmark datasets and compare it with state-of-the-art competitors in the classification task.

Experimental Setting. We consider KNN, PIVOTTREE (PT) [8], and ϵBALL [5] as baseline instance-based models, all using the Euclidean distance as d. Next, we evaluate PDTF against these baselines by integrating such classifiers in the framework, namely PDT-KNN, PDT-PT, and PDT-ϵBALL using PDT as distance function d. Additionally, we compare PDTF with the pairwise distance classifier PDL [2] to provide a comprehensive performance assessment. Specifically, we consider KNN using Euclidean distance with PDL, namely PDL-KNN. For KNN[6] we use $k = 5$, for PT[7] we use $maxdepth = 5$, and for ϵBALL[8] we use ϵ equals to the 10^{th} percentile of all pairwise distances.

We measure the classification performance using the Accuracy and the weighted F1-score [34]. We assess the effectiveness of the PDT as a regressor by reporting the R^2 and $RMSE$ score [34], which quantifies its ability to approximate true pairwise distances. For each experiment, an 80/20 train/test split is applied. Results are reported on the test set. Finally, we evaluate the computational efficiency by measuring both training and prediction times, reported in seconds.

As datasets, we use two sets of tabular benchmark datasets. First, for the sensitivity analysis, we follow the OpenML-CC18 [6] selection constraints and select 9 relatively `small` datasets from OpenML[9]. These datasets are employed in other papers of PSL and PDL [2], making them particularly suitable for our approach. Next, for comparisons against competitors, we rely on 11 datasets from OpenML and other repositories[10], e.g. `spambase` (UCI), `compas` (ProPublica).

[5] Python implementation available at: `https://github.com/fismimosa/PDT`.

[6] KNN as implemented in `sklearn`: `https://tinyurl.com/sklearn-knn`.

[7] PIVOTTREE as implemented in: `https://github.com/msetzu/pivottree`.

[8] ϵBALL as implemented in: `https://tinyurl.com/epsball`.

[9] The `small` datasets are: iris, seeds, glass, algerian forest fires (fire), vertebra column (verteb), ecoli, low resolution spectrometer (lrs), breast cancer (breast).

[10] The `large` datasets are: steel plates fault (steel), read safety (road), bank marketing (bank), pol, covertype (cover), house 16H (house), eye movements (eye), sylvine (sylvine), magic telescope (magic), compas, spambase (spam).

Table 2: Mean and std.dev of R^2 and *RMSE* for PDT as distance regressor among the `small` datasets, with $\tilde{n}=n$, i.e., without sampling strategy, and *maxdepth*=8, with different pairwise training set preparation (α, β, γ) and different split constraints (P, F, T). Best in bold, second best in italic.

PDT	R^2			*RMSE*		
	P	F	T	P	F	T
α	.660 ± .138	.652 ± .137	.631 ± .133	1.53 ± .845	1.55 ± .870	1.60 ± .890
β	**.909 ± .056**	**.909 ± .056**	**.909 ± .056**	**.830 ± .575**	**.830 ± .575**	**.830 ± .575**
γ	*.907 ± .059*	*.907 ± .059*	*.907 ± .059*	*.836 ± .580*	*.837 ± .581*	*.837 ± .581*

Table 3: Mean and std.dev of Accuracy and of weighted F1-score for PDTF using KNN as classifier among `small` datasets, with $\tilde{n}=n$, i.e., without sampling strategy, and *maxdepth*=8, with different pairwise training set preparation (α, β, γ) and different split constraints (P, F, T). Best in bold, second best in italic.

PDT	Accuracy			F1-score		
	P	F	T	P	F	T
α	.700 ± .098	.697 ± .128	.662 ± .183	.654 ± .126	.656 ± .161	.607 ± .210
β	*.808 ± .133*	*.808 ± .133*	*.808 ± .133*	*.799 ± .143*	*.799 ± .143*	*.799 ± .143*
γ	**.809 ± .133**	**.809 ± .133**	**.809 ± .133**	**.800 ± .144**	**.800 ± .144**	**.800 ± .144**

Detailed summaries of each dataset, including the number of records, features (after removing the categorical[11] ones), and classes, are provided in Table 1.

Sensitivity Analysis. We analyze here how the hyper-parameters of PDTF influence both the quality of distance approximation and the classification performance. Table 2 presents the mean and std. dev. of the R^2 and *RMSE* score for PDT, which assesses its effectiveness as a distance approximation using a regression evaluation measure, while Table 3 shows the Accuracy and weighted F1-score for PDTF with KNN across the `small` datasets using KNN as classifier. In this initial analysis, we examine the impact of different pairwise training set preparations (α, β, γ) and split constraints (P, F, T) while fixing $\tilde{n} = n$, i.e., without applying any sampling strategy, and setting *maxdepth*=8, a relatively compact tree that balances interpretability and performance.

The results highlight the importance of the pairing procedure in PDTF. Simply using concatenation (α) leads to higher *RMSE* error and lower performance for R^2, Accuracy and F1-score. However, adding pointwise feature difference (β) or combining it with concatenation (γ) significantly improves distance approximation and classification performance when used in KNN. The performance difference between γ and β is minimal. Analysis of the tree structures in the γ

[11] The PDTF framework remains fully applicable to mixed-type data, computing the target PDT mixed-distance for each pair by combining a numerical metric on continuous attributes with a categorical metric on nominal attributes.

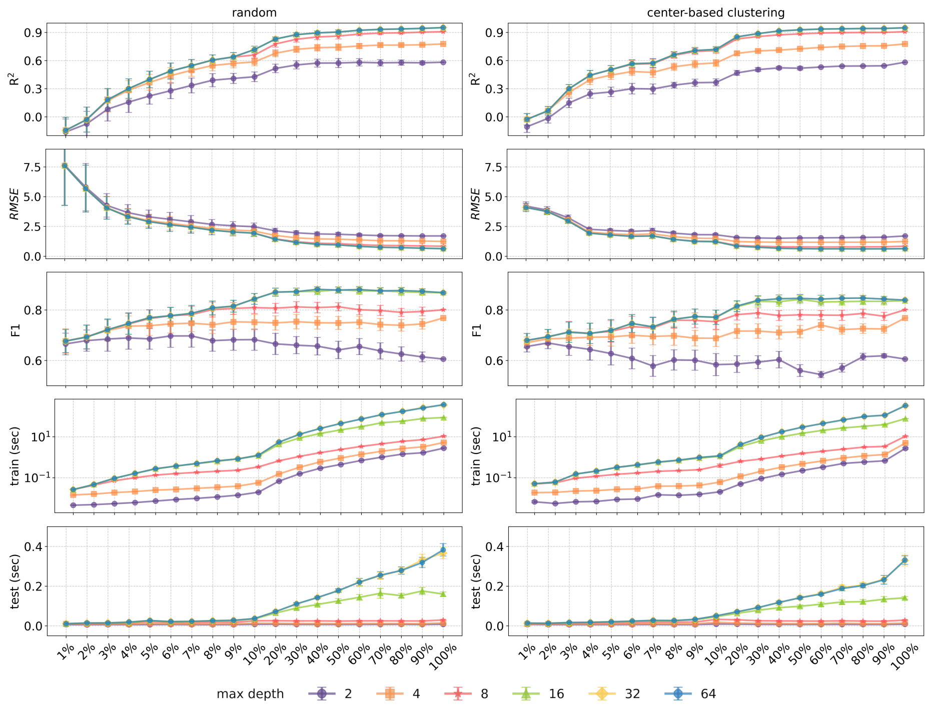

Fig. 3: Errorbars for PDT-γT with varying $\tilde{n}$ and *maxdepth* using the random and center-based clustering sampling strategies over the `small` datasets.

setting shows that pointwise features are preferred over concatenated ones, explaining why split constraints impact only α and γ. In the α setting, increasing split constraints degrades all metrics, suggesting a trade-off between distance approximation and interpretability. Based on these results, PDT-γT is recommended as the default configuration. This setup benefits from including both feature types, allowing the model to optimally adapt by selecting the most informative features during training. Also, PDT-T shows strong generalization and offers the highest interpretability.

Next, we studied the impact of the *maximum depth* of PDT by varying it in $\{2^i \mid i \in \mathbb{Z}, 1 \leq i \leq 6\}$, and the impact of the random sampling (RS) and center-based clustering sampling (CS) by varying $\tilde{n} \in \{n \cdot i\% \mid i \in \mathbb{Z}, 0 \leq i \leq 10\} \cup \{n \cdot 10i\% \mid i \in \mathbb{Z}, 0 \leq i \leq 10\}$, with each experiment repeated ten times to compute mean performances and standard deviations.

The results, illustrated in the errorbars[12] of Figure 3, show that using approximately 20% of the training dataset ($\tilde{n} \approx 20\% \cdot n$) for creating the pairwise training set leads to a performance plateau for both R^2, *RMSE*, and F1, regardless of the sampling strategy, except for the smallest tree depth. In this case, F1

[12] For the sake of comprehensibility we reduced the std to 0.5 of its value.

Table 4: Mean and std.dev of weighted F1-score, Accuracy, Training Time, and Prediction Time expressed in seconds over all datasets. The best and second best are highlighted in bold and italics, respectively, by column.

	model	d	F1-score	Accuracy	Train Time	Pred Time
`small`	KNN	EUC	**.868 ± .118**	**.877 ± .099**	**0.0001 ± 0.0**	**0.0001 ± 0.0**
		PDT$_8$	.790 ± .130	.799 ± .117	0.471 ± 0.446	0.016 ± 0.015
		PDT$_{16}$	.838 ± .098	.842 ± .091	3.040 ± 2.640	0.051 ± 0.045
		PDL	*.845 ± .096*	*.863 ± .078*	0.344 ± 0.657	39.30 ± 69.99
	PT	EUC	.828 ± .126	.840 ± .108	0.067 ± 0.032	*0.002 ± 0.000*
		PDT$_8$	.814 ± .091	.821 ± .084	9.380 ± 7.510	0.063 ± 0.040
		PDT$_{16}$	.819 ± .091	.825 ± .008	16.780 ± 14.46	0.115 ± 0.077
	ϵBALL	EUC	.833 ± .009	.842 ± .008	*0.01 ± 0.005*	0.041 ± 0.027
		PDT$_8$	.673 ± .204	.710 ± .162	0.665 ± 0.608	0.232 ± 0.150
		PDT$_{16}$	.691 ± .149	.730 ± .125	1.206 ± 1.257	0.545 ± 0.508
`large`	KNN	EUC	**.753 ± .124**	**.756 ± .125**	**0.0001 ± 0.0**	**0.002 ± 0.0**
		PDT$_8$	.597 ± .119	.608 ± .118	3.860 ± 2.030	0.232 ± 0.089
		PDT$_{16}$	.670 ± .112	.677 ± .113	31.470 ± 16.220	0.546 ± 0.140
	PT	EUC	.704 ± .124	.715 ± .122	1.444 ± 1.453	*0.005 ± 0.0*
		PDT$_8$	*.713 ± .128*	*.719 ± .127*	445.37 ± 450.76	0.654 ± 0.197
		PDT$_{16}$	.713 ± .132	.718 ± .131	841.45 ± 742.11	1.400 ± 0.261
	ϵBALL	EUC	.709 ± .137	.717 ± .134	*0.231 ± 0.286*	0.03 ± 0.019
		PDT$_8$	.467 ± .170	.537 ± .117	20.322 ± 7.372	2.571 ± 1.321
		PDT$_{16}$	.606 ± .123	.631 ± .105	39.544 ± 12.632	10.691 ± 6.313

slightly decreases as more data is used. Higher tree depths generally reduce performance variability. While random and center-based sampling strategies yield similar R^2 and *RMSE* values, random sampling results in better overall F1 when integrated with KNN. Increasing tree depth beyond 16 does not provide significant performance gains, making $maxdepth = 16$ a good trade-off between performance and interpretability. A shallower tree with $maxdepth = 8$ maintains similar regression performance with minimal weighted F1-score loss. Training and prediction times increase with larger datasets and deeper trees, but trees with depth 16 are faster than those with depths of 32 or 64, and depth 8 is the fastest. Based on these findings, we recommend the PDT-γT variant with $\tilde{n} \approx 20\% n$ and $maxdepth = 16$ as the baseline configuration, providing a balanced approach to performance, efficiency, and interpretability. Reducing tree depth to 8 improves efficiency with only a slight performance trade-off.

Competitor Analysis. In Table 4 we analyze the performance of PDTF against competing methods on both the `small` and `large` datasets. Specifically, we compare the standard Euclidean distance function and PDL against two variants of PDT with maximum depths of 8 and 16 (PDT$_8$ and PDT$_{16}$), as suggested by our earlier discussion on tree depth. For a comprehensive evaluation, we adopt

Table 5: Mean and std.dev of R^2 and *RMSE* for KNN model with PDT suggested depths on `small` and `large` datasets. Best by dataset size highlighted in bold.

	d	R^2	*RMSE*
`small`	PDT_8	$0\ .764 \pm 0.214$	1.559 ± 1.432
	PDT_{16}	$\mathbf{0.775 \pm 0.217}$	$\mathbf{1.530 \pm 1.439}$
`large`	PDT_8	0.601 ± 0.307	2.642 ± 2.985
	PDT_{16}	$\mathbf{0.670 \pm 0.320}$	$\mathbf{2.455 \pm 3.023}$

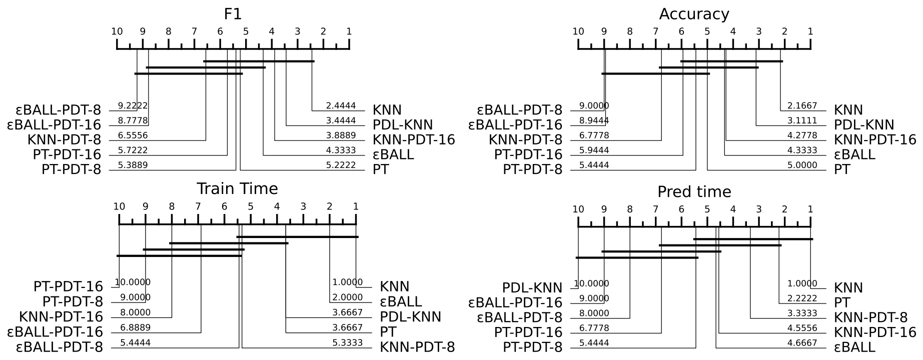

Fig. 4: Comparison of model's rank for the various evaluation measures against each other with the Nemenyi test. Groups of classifiers that are not significantly different at 95% significance level are connected. Best ranks on the right.

the aforementioned instance-based models: KNN, PT, and ϵBALL[13]. We report results for PDT variants using a fixed random subset for approximation, with $\tilde{n} \approx 20\% n$ for all datasets except `spambase` and `compas`, where $\tilde{n} \approx 5\% n$. The comparison of the ranks of all methods tested on the `small` datasets[14] is visually represented through the critical difference plots in Figure 4. Methods that are statistically equivalent, according to the post-hoc Nemenyi test, are connected by black lines. Lower rank values correspond to better-performing models, with the best ranks displayed on the right (see [12] for details).

Results on the `small` datasets show that PDT achieves F1 and Accuracy scores comparable to its competitors, though slightly lower. However, these differences are not statistically significant, as illustrated in Figure 4. In terms of runtime, PDT outperforms PDL significantly, both during inference and in terms of training and prediction time, while offering full interpretability. We also note that the expected runtime of PDT is lower than that of KNN at prediction time when $\log \tilde{n} < m$, but deviations from theoretical expectations are observed due to the current implementation. While we acknowledge that empirical time mea-

[13] The ϵBALL strategy is solely used to select memory instances based on different distance metrics, while the final classification is performed using KNN with $k = 5$.

[14] Similar results are obtained for `large` datasets but not reported due to lack of space.

		mean rad.	mean per.	mean area	mean conc.	mean conc. p.	area err.	worst rad.	worst per.	worst area	worst conc. p.	distance	class
	x	3.84	3.98	5.44	3.29	3.09	10.84	4.30	4.52	6.58	2.16	–	M
PDT	x_1	3.34	3.44	3.99	2.97	3.67	2.41	3.66	3.83	4.59	2.36	1.90	M
	x_2	3.02	3.07	3.49	1.68	2.52	2.12	2.96	3.08	3.43	1.94	3.63	M
	x_3	3.19	3.33	3.60	2.93	3.49	1.69	2.97	3.27	3.27	2.52	3.63	M
EUC	x_1	4.04	4.05	5.44	2.76	2.84	10.48	2.56	2.54	3.15	0.64	4.64	M
	x_2	3.34	3.44	3.99	2.97	3.67	2.41	3.66	3.83	4.59	2.36	8.90	M
	x_3	2.91	3.10	3.25	4.06	3.92	4.14	2.10	2.30	2.32	1.61	8.99	M

```
IF worst_perimeter_diff <= 1.48 AND worst_perimeter_diff <= 0.71
AND mean_concavity_diff <= 1.12 AND mean_area_diff > 0.45 THEN 1.91

IF worst_perimeter_diff <= 1.48 AND worst_perimeter_diff > 0.71
AND mean_concave_points_diff <= 1.48 AND mean_area_diff > 0.99 THEN 3.63
```

Fig. 5: Classification example for the instance x w.r.t 3NN using PDT and EUCLIDEAN distances. Predicted class is Malignant. PDT rules are under the table.

surements may be affected by implementation-specific factors, we have retained them as an integral component of our evaluation, since they highlight practical considerations that are essential for understanding PDTF's overall behavior. Future work will focus on improving the implementation of PDTF to better align with theoretical complexity. Additionally, PDL results for the `large` datasets are excluded, as even the smallest large dataset failed to produce results within a 24-hour runtime, a limitation shared with PT. For the `large` datasets, KNN with Euclidean distance remains the best performer across all metrics. However, we note that PDT-PT with *maxdepth* = 8 outperforms PT with the standard Euclidean distance and PDT-KNN. This combination provides the dual advantage of an intelligent *pivot* selection process coupled with the use of an interpretable distance function to make the final decision. For ϵBALL, the performance of PDT always deteriorates compared to EUC. Finally, Table 5 reports the R and *RMSE* regression metrics for PDT$_8$ and PDT$_{16}$ paired with KNN. These results echo the trend from Table 4: larger datasets present a uniformly harder task than smaller ones, regardless of the model, and this increased difficulty is reflected in the regression metrics where applicable. Consequently, the drops in F1-score and Accuracy observed can be attributed to the regressor tree's reduced ability to approximate distances accurately under these more challenging conditions.

Explanatory Example. Figure 5 presents an example of KNN with $k = 3$ on the `breast cancer` dataset to classify cells as *Benign* or *Malignant*, using both Euclidean distance and PDT[15]. The first row displays the query instance x. The subsequent six rows list the three nearest neighbors selected by KNN for both distance measures. The distances are shown in the penultimate column, and the classes are shown in the last column. The key advantage of PDT over Euclidean distance is that with PDT, it is possible to inspect the decision rules under the table, which logically justify the calculated distances. As shown by our experimental evaluation, the approximation error of the learned PDT is acceptably small in its traded off for a fully transparent-by-design model structure. In contrast, post-hoc explainability techniques are applied only after training and can themselves introduce artifacts into the explanation [29]. PDTF avoids these pitfalls and yields rule-based explanations that directly justify pairwise similarity decisions, rather than merely attributing feature importance to a final classification outcome. Additionally, the rule-based explanations are concise and involve only the subset of features that actually contribute to the computed distance, rather than requiring inspection of all features. Finally, while Euclidean distance is often considered intuitive, its interpretability becomes less clear as the number of features increases. For example, consider two records x and x_1 described by 10 attributes a_1 to a_{10}. Suppose their pairwise differences are: $x[a_1] - x_1[a_1] = 0.50$, $x[a_2] - x_1[a_2] = 0.54$, $x[a_3] - x_1[a_3] = 1.45$, ..., $x[a_{10}] - x_1[a_{10}] = -0.20$. The Euclidean distance in this case is the square root of the sum of all squared differences, resulting in a single value that blends together contributions from all features. While each individual difference is easy to interpret, the final distance value $\sqrt{(0.50)^2 + (0.54)^2 + \cdots + (-0.20)^2}$ does not directly reveal which attributes were most responsible for the similarity or dissimilarity between the records. This makes it difficult to extract a simple, human-understandable explanation for why x is considered close to x_1.

5 Conclusion

We have presented Pairwise Distance Tree Framework (PDTF), an interpretable meta-learning approach designed to enhance transparency in instance-based models. It replaces traditional complex distance functions with a shallow PDT regressor, which learns a mapping from instance pairs to their respective distances. By combining the strengths of Pairwise Distance Learning (PDL) and Proxy-based Similarity Learning (PSL), PDTF offers both efficient instance selection and clear, traceable decision rules. Experimental results on benchmark datasets show that PDTF strikes a strong balance between predictive performance, computational efficiency, and interpretability. It outperforms traditional methods, especially when intelligent sampling is used, reducing training time without compromising accuracy. Enforcing forced split constraints further enhances interpretability, though it may slightly impact performance. PDTF is highly customizable, with adjustable PDT depth, making it adaptable to different

[15] Due to space limitations, this example is restricted to the 10 most important features of the dataset, as shown at this `sklearn` link: `https://tinyurl.com/breast-features`.

applications. Future work will focus on jointly optimizing distance metrics and instance selection, and on evaluating the method's robustness in zero-shot classification. We also plan to extend PDTF to other modalities, such as images and time series, by leveraging inherently interpretable features. Doing so presents its own challenges and will require a dedicated study, since feature interpretability in these domains remains sparsely addressed. We also plan to optimize the PDTF implementation to bridge the gap between empirical computational times and theoretical expectations. Finally, we would like to conduct an extrinsic interpretability evaluation of PDTF usage through a human decision-making task driven by its explanations. Overall, PDTF lays the foundation for developing transparent and efficient instance-based models across diverse domains.

Acknowledgments. This work has been partially supported by the Italian Project Fondo Italiano per la Scienza FIS00001966 "MIMOSA", by the European Community Horizon 2020 programme under the funding schemes ERC-2018-ADG G.A. 834756 "XAI", G.A. 101070212 "FINDHR", G.A. 101120763 "TANGO", by the European Commission under the NextGeneration EU programme – National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR) Project: "SoBigData.it – Strengthening the Italian RI for Social Mining and Big Data Analytics" – Prot. IR0000013 – Av. n. 3264 del 28/12/2021, and M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR" - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI".

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aha, D.W.: The omnipresence of case-based reasoning in science and application. Knowl. Based Syst. **11**(5-6), 261–273 (1998)
2. Belaid, M.K., et al.: Pairwise difference learning for classification. In: DS, Part II. LNCS, vol. 15244, pp. 284–299. Springer (2024)
3. Bichindaritz, I., Marling, C.: Case-based reasoning in the health sciences: What's next? Artif. Intell. Medicine **36**(2), 127–135 (2006)
4. Bien, J., Tibshirani, R.: Hierarchical clustering with prototypes via minimax linkage. J. Amer. Statist. Assoc. **106**(495), 1075–1084 (2011)
5. Bien, J., Tibshirani, R.: Prototype selection for interpretable classification (2011)
6. Bischl, B., et al.: Openml benchmarking suites. In: NeurIPS (2021)
7. Breiman, L., et al.: Classification and regression trees. Routledge (2017)
8. Cascione, A., et al.: Data-agnostic pivotal instances selection for decision-making models. In: ECML PKDD, Part I. LNCS, vol. 14941, pp. 367–386. Springer (2024)
9. Chen, Y., et al.: Benchmark of general-purpose machine learning-based quantum mechanical method AIQM1. JCP **158**(7), 074103 (02 2023)
10. Cherkauer, K.J., Shavlik, J.W.: Growing simpler decision trees to facilitate knowledge discovery. In: KDD. pp. 315–318. AAAI Press (1996)
11. Corbara, S., et al.: Same or different? diff-vectors for authorship analysis. ACM Trans. Knowl. Discov. Data **18**(1), 12:1–12:36 (2024)
12. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR (2006)

13. Deng, J., et al.: Arcface: Additive angular margin loss for deep face recognition. vol. 44, pp. 5962–5979 (2022)
14. Domingos, P.M.: The role of occam's razor in knowledge discovery. Data Min. Knowl. Discov. **3**(4), 409–425 (1999)
15. Endou, T., Zhao, Q.: Generation of comprehensible decision trees through evolution of training data. In: CEC. pp. 1221–1225. IEEE (2002)
16. Fix, E., et al.: Discriminatory analysis, nonparametric discrimination (1951)
17. Freitas, A.A.: Comprehensible classification models: a position paper. SIGKDD Explor. **15**(1), 1–10 (2013)
18. Gao, X.W., Gao, A.: COVID-CBR: A deep learning architecture featuring case-based reasoning for classification of COVID-19 from chest x-ray images. In: ICMLA. pp. 1319–1324. IEEE (2021)
19. Guidotti, R., et al.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 93:1–93:42 (2019)
20. Hu, J., et al.: Exploring a general convolutional neural network-based prediction model for critical casting diameter of metallic glasses. JAC **947**, 169479 (2023)
21. Hu, J., et al.: Discriminative deep metric learning for face verification in the wild. In: CVPR. pp. 1875–1882. IEEE Computer Society (2014)
22. Johnson-Laird, P.N.: Mental models and human reasoning. PNAS **107**(43), 18243–18250 (2010). https://doi.org/10.1073/pnas.1012933107
23. Kaya, M., et al.: Deep metric learning: A survey. Symmetry **11**(9), 1066 (2019)
24. Li, W., et al.: A data-driven explainable case-based reasoning approach for financial risk detection. Quant. Finance **22**(12), 2257–2274 (2022)
25. Liu, W., et al.: Sphereface: Deep hypersphere embedding for face recognition. In: CVPR. pp. 6738–6746. IEEE Computer Society (2017)
26. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
27. Quinlan, J.R.: Programs for Machine Learning C4. 5. Elsevier (1993)
28. Rokach, L., Maimon, O.: Top-down induction of decision trees classifiers - a survey. IEEE Trans. Syst. Man Cybern. Part C **35**(4), 476–487 (2005)
29. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat. Mach. Intell. **1**(5), 206–215 (2019)
30. Sasikumar, M.: Case Based Reasoning, vol. 8 (1998)
31. Schank, R.C., Abelson, R.P.: Knowledge and memory: The real story. In: Knowledge and memory: The real story, pp. 1–85. Psychology Press (2014)
32. Slade, S.: Case-based reasoning: A research paradigm. AI Mag. **12**(1), 42–55 (1991)
33. Spelke, E.: What babies know: Core knowledge and composition volume 1, vol. 1. Oxford University Press (2022)
34. Tan, P., Steinbach, M.S., Kumar, V.: Introduction to data mining (2005)
35. Tynes, M., et al.: Pairwise difference regression: A machine learning meta-algorithm for improved prediction and uncertainty quantification in chemical search. J. Chem. Inf. Model. **61**(8), 3846–3857 (2021)
36. Wang, Y., King, R.D.: Extrapolation is not the same as interpolation. Mach. Learn. **113**(10), 8205–8232 (2024)
37. Wen, Y., et al.: Pairwise similarity learning is simple. In: ICCV 2023,. pp. 5285–5295. IEEE (2023)
38. Wetzel, S.J., et al.: Twin neural network regression is a semi-supervised regression algorithm. Mach. Learn. Sci. Technol. **3**(4), 45007 (2022)
39. Yang, L., Jin, R.: Distance metric learning: A comprehensive survey. Michigan State Universiy **2**(2), 4 (2006)
40. Yao, X., et al.: Adaptive deep metric learning for affective image retrieval and classification. IEEE Trans. Multim. **23**, 1640–1653 (2021)