Data-Agnostic Pivotal Instances Selection for Decision-Making Models

Alessio Cascione[1], Mattia Setzu[1], and Riccardo Guidotti[1,2]

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

Abstract. As decision-making processes become increasingly complex, machine learning tools have become essential resources for tackling business and social issues. However, many methodologies rely on complex models that experts and everyday users cannot really interpret or understand. This is why constructing interpretable models is crucial. Humans typically make decisions by comparing the case at hand with a few exemplary and representative cases imprinted in their minds. Our objective is to design an approach that can select such exemplary cases, which we call *pivots*, to build an interpretable predictive model. To this aim, we propose a hierarchical and interpretable pivot selection model inspired by Decision Trees, and based on the similarity between pivots and input instances. Such a model can be used both as a pivot selection method, and as a standalone predictive model. By design, our proposal can be applied to any data type, as we can exploit pre-trained networks for data transformation. Through experiments on various datasets of tabular data, texts, images, and time series, we have demonstrated the superiority of our proposal compared to naive alternatives and state-of-the-art instance selectors, while minimizing the model complexity, i.e., the number of pivots identified.

Keywords: Interpretable Machine Learning · Explainable AI · Instance-based Approach · Pivotal Instances · Transparent Model

1 Introduction

In recent years, Machine Learning (ML) models have become increasingly central in supporting human decision-making processes [11]. These models are relied upon to tackle business problems and social issues in health science, online threat detection, and shopping pattern analysis [9, 14, 21], among others. Still, these models rely on complex architectures, making it difficult for anyone, experts and end users alike, to understand their reasoning. Moreover, although these tools may achieve identical or even superior performances compared to humans, the "cognitive process" they employ is hardly comparable to the one humans may use to solve the same task [43]. Given the pervasiveness of these models, interpreting and explaining their predictions and decisions generated, ultimately

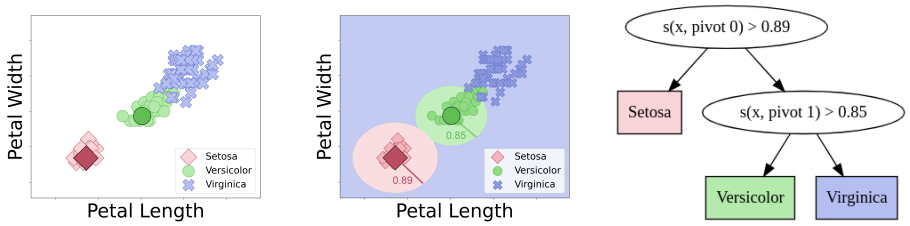
unveiling the internal mechanism inside the "black-box", is crucial [27]. We can identify this as the main goal of Explainable AI (XAI) [7].

In order to construct ML models that are inherently interpretable, a possible avenue to explore involves harnessing the intuitive notion of similarity of *discriminative* or *descriptive* elements. Our fundamental assumption is that a model "reasoning" in terms of exemplary instances provides an inherently interpretable tool to decision-makers, analysts, and end-users alike [41]. As humans, our cognitive processes and mental models often rely on a form of *case-based reasoning* [38] in which we store in our memory a large set of past exemplary cases, and then retrieve them as needed according to the task at hand. While the retrieval mechanism is itself obscure, reasoning in terms of said similar cases is inherently interpretable. This form of reasoning is so ingrained in us that even small children are able to recognize, use, and play with novel objects they have never seen, but that, in some form, are similar to other objects that they already know [39]. Furthermore, this applies to a wide variety of modalities: we recognize relatives based on faces we have already seen, music genres and bands based on song we have already heard, the origin of a recipe based on other recipes we have already tasted, etc [26]. At its most fundamental level, similarity, and more generally case-based reasoning, is a universal form of human reasoning, pervasive to a plethora of modalities and data types [19].

Case-based reasoning offers significant advantages for fostering interpretability across various domains such as health [4], financial risk prediction [31], general text domains [12, 24], and time-series and image analysis [1]. Particularly in the latter, recent research [25, 36] shows good promise on the effectiveness of this type of reasoning, which is often preferred by human subjects. Given these premises, we emphasize the importance of training data quality as a ground for similarity between pivots and instances to predict: poor diversity or bias can result in unrepresentative cases. In contrast, feature-based methods may be more robust in such contexts due to their focus on how features influence outcomes.

This paper aims to design an interpretable case-based model that selects descriptive and discriminative cases to solve a decision-making task. With this in mind, we introduce PivotTree, a hierarchical and interpretable case-based model inspired by Decision Trees [8]. By design, PivotTree lends itself to both *selection* and *prediction*. As a selection model, PivotTree identifies a set of *pivots*, exemplary cases identified within a training set. As a predictive model, PivotTree leverages the selected pivots to build a similarity-based Decision Tree, routing instances through its structure, and yielding a prediction and an associated explanation. Unlike traditional Decision Trees, the explanation is not a set of rules, but rather a set of pivots to which the instance is similar. Like distance-based models, PivotTree is also a selection method, encoding instances in a similarity space that enables case-based reasoning. Finally, PivotTree is a *data-agnostic* model, which can be applied to different data modalities, jointly solving both pivot selection and prediction tasks.

Figure 1 provides an example of PivotTree on the `iris` dataset, wherein flowers are classified according to their petal characteristics. Starting from a



(a) Select (b) Predict (c) Explain

Fig. 1: PivotTree as *(a)* selector, *(b)* interpretable model, *(c)* Decision Tree.

dataset of instances, PivotTree filters down a set of pivots (Figure 1 *(a)*), i.e., a set of representative flowers. Said *pivots* are then used to learn a case-based model wherein novel instances are represented in terms of their similarity to the induced pivots (Figure 1 *(b)*). Building on pivot selection, PivotTree then learns a hierarchy of pivots wherein instances are classified. This hierarchy takes the form of a Decision Tree (Figure 1 *(c)*): novel instances navigate the tree, percolating towards pivots to which they are more similar, ultimately building a chain of similar pivots, and landing in a classification leaf. In this case, given a test instance x: if its similarity to *pivot 0* is higher than 0.89 (following the left branch), then x is classified as a *Setosa* flower. Otherwise (following the right branch), if x's similarity to *pivot 1* is higher than 0.85 (left branch), then x is classified as a *pivot 0* flower. If neither condition is met, x is classified as a *Virginica* flower. In contrast, a traditional Decision Tree (DT) would model the decision boundary with feature-based rules, e.g., "if petal length $<$ 2.4 then Setosa else if petal width $<$ 1.7 then Versicolor else Virginica". However, *(i)* such an approach can only model axis-parallel splits, and *(ii)* cannot be employed on data types with features without clear semantics. Hence, improving on traditional DTs, the case-based model learned by PivotTree can provide interpretability even in domains such as images, text, and time series, where by-design interpretable models are both underperforming and lack interpretability. Furthermore, unlike conventional state-of-the-art distance-based predictive models such as kNN [17], our proposal introduces a hierarchical structure to guide similarity-based predictions.

Experiments conducted on 24 datasets of different modalities, i.e., tabular data, time series, images, and text, show that PivotTree yields interpretable predictive models that are as effective as state-of-the-art approaches at a fraction of their complexity expressed as the number of pivots. Qualitative results indicate high effectiveness on different data modalities, while a sensitivity analysis shows stability in the accuracy when varying the number of pivots selected.

After a review of some related works in Section 2, in Section 3 we illustrate our proposal. Then, Section 4 reports the experimental results. Finally, Section 5 summarizes our contributions and open research directions.

2 Related Work

Similarity-based methods belong to one of two families: *similarity* methods, aiming to, given a fixed data representation, learn the proper pivots[3] through similarity on said representation; and *representation* methods, which instead fix a similarity function, and aim to learn a proper instance representation.

Similarity. Underlying similarity methods is the assumption of a fixed data representation. Among them, we can distinguish three subclasses of methods: *covering*, *clustering*, and *partitioning* methods. *Covering* methods aim to group records around pivots. ε-BALL [6] jointly learns a set of distance-based neighboring *coverages* centered on a set of pivots. Pivots are optimized to be as few as possible, while coverages to be as class-pure as possible. The resulting pivots are thus laid on a "flat" structure where no structure defines the relationship *among* pivots. *Clustering* algorithms can provide more nuanced pivot-to-pivot relationships by tackling the lack of inter-pivot relationships. The MINIMAX algorithm [5] builds on agglomerative clustering by identifying cluster representatives, aggregating them in a hierarchical fashion, resulting in a hierarchy of prototypes. PIVOTTREE improves on MINIMAX by greatly improving on its complexity, and by leveraging pivots to perform prediction. *Partitioning* algorithms segment the feature space, assigning a pivot to each segment. PROXIMITYFOREST [32] induces a forest of similarity-based Decision Trees routing instances according to two pivots similarities. Notably, pivots are selected randomly, and so is the similarity function, thus yielding highly randomized trees. Unlike covering algorithms, PIVOTTREE constructs hierarchies of pivots, thus improving model interpretability. Like partitioning algorithms, PIVOTTREE partitions the feature space, but unlike PROXIMITYFOREST, it adopts a pivot selection strategy and a fixed similarity function, greatly improving the robustness and variance of its results. Finally, in [45] is presented a related methodology to select the best split for DTs based on the average similarity of instance pairs belonging to each children node. While being comparable to PIVOTTREE as they both determine the best split w.r.t. a similarity function, despite the title, they are inherently different as in [45] are not identified prototypical instances, using traditional feature-based rules for the split.

Representation. Unlike similarity methods, *representation methods* fix a similarity function, and rely on learning a proper representation of the data to find pivots. Unsurprisingly, these methods are often neural models lacking interpretability. [18] and [35] introduce soft Decision Trees, wherein nodes hold pivots, and instances are routed probabilistically towards multiple paths in the tree, thus creating fuzzy chains of pivots. Other approaches improve on the data representation at the cost of the intra-pivot structure. The authors in [2] introduce a neural model that jointly learns the data representation and a set of pivots, which are later used for classification. Similarly, PROTOPNET [10] and HPNET [22] learn a neural network that identifies pivots by learning contrastive

[3] In Section 2 we adopt the term *pivot* to refer to the instances selected by different proposals in the state-of-the-art which do not necessarily adopt this term.

representations and employing them for classification. Recently, a set of extensions of ProtoPNet have been proposed. ProtoTEx [12] integrates a similar approach for texts using pre-trained language models. ProtoSeNet [33] offers a model where pivots can be refined through user knowledge for general sequence-based data. ProtoryNet [24] improves ProtoSeNet by handling longer textual sequences. Finally, by providing even more fine-grained pivots, CNN-Trees [44] learn a neural model that constructs a hierarchy of pivots, each layer more specific than the previous, and each pivot also providing a score indicating its contribution to the final prediction. Unlike neural representation methods, PivotTree learns a crisp and fully interpretable model.

3 Pivot Tree Selection Model

We present here PivotTree, an interpretable hierarchical pivot selection model inspired by Decision Trees [8]. Let us start by formalizing the problem and setting. Without loss of generality, we restrict ourselves to classification tasks and leave other tasks for future work.

Problem Setting. Given a population of instances represented as real-valued m-dimensional feature vectors[4] in $\mathbb{R}^m$ and a set of class labels $C = \{1, \ldots, c\}$, we assume the existence of an unknown ground-truth function $g : \mathbb{R}^{n \times m} \to C$ mapping each vector in $\mathbb{R}^m$ to one of the c classes in C. In case-based reasoning, the objective is to learn a function $f : \mathbb{R}^m \to C$ approximating g, with f being defined as a function of k exemplary cases named *pivots*. As explained in Sec. 2, similarity-based case-based models define f on a similarity space, often inversely denoted as "distance space", $\mathcal{S}$ induced by a similarity function $s : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ quantifying the similarity of instances [37].

Given a training set $\langle X, Y \rangle$ with $X = \{x_i\}_{i=1}^n$ of n instances, $Y = \{y_i\}_{i=1}^n$ with $y_i \in C$ the associated class labels, and a similarity function s, our objective is to learn a function $\pi : \mathbb{R}^{n \times m} \to \mathbb{R}^{k \times m}$ that takes as input X and returns a set $P \subseteq X$, i.e., $\pi_s(X) = P$, of k pivots such that the performance of f are maximized. Furthermore, aiming for transparency of the case-based predictive model f, our objective is to employ as an interpretable model f Decision Tree classifiers or k-Nearest Neighbors approaches [20] (kNN).

In practical terms, given a training set $\langle X, Y \rangle$ and a similarity function s, the selection method π selects k pivots P from X. Through the similarity function s and the pivots P, the dataset $X \in \mathbb{R}^m$ is mapped into the *similarity space* $\mathcal{S}$, and thus encoded into a representation $Z \in \mathbb{R}^{n \times k}$ where $Z_{i,j}$ is the similarity between the i-th object with the j-th pivot in P. Hence, the predictive model f is trained on $\langle Z, Y \rangle$. Then, given a test instance $x \in \mathbb{R}^m$, x is first mapped to a similarity vector $z = \langle s(x, p_1), \ldots, s(x, p_k) \rangle$ yielding its similarity to the set P of pivots. Then, z is provided to f, which performs the prediction.

[4] For the sake of simplicity, we consistently treat data instances as real-valued vectors. Any data transformation employed in the experimental section to maintain coherence with this assumption will be specified when needed.

When f is implemented with a Decision Tree, split conditions will be of the form $s(x, p_i) \geq \beta$, i.e., "if the similarity between instance x and pivot p_i is greater or equal then β, then ...", allowing to easily understand the logic condition by inspecting x and p_i for every condition in the rule.

On the other hand, when f is implemented as a kNN, every decision will be based on the similarity with a few neighbors (typically between one and five) in the similarity space $\mathcal{S}$ obtained computing the similarity between each instance with respect to the selected pivots. A human user just needs to inspect x and the similarities with the pivots P and the instances in the neighborhood. When the number of pivots is kept small, the interpretability of both methods increases, limiting the expressiveness. Vice versa, using a selection model π that returns a large number k of pivots can increase the performance at the cost of interpretability. Our proposal aims to balance these two aspects by allowing the selection of a small number of pivots that still guarantee comparable performance to interpretable predictive models.

PivotTree Algorithm. In this paper, with PivotTree, we implement the selection function π. Much like Decision Tree induction algorithms [8], PivotTree greedily learns a hierarchy of nodes, each node splitting instances towards one of its two children, ultimately reaching terminal leaf nodes, which are associated with a classification label. The splitting is based on *discriminative* pivots and *representative* pivots. Let X_t be the records constrained by the decision path at iteration t in the tree construction, and Y_t the associated class labels, then a discriminative pivot is an instance of class c, i.e., $p^- \in X_t^{(c)} = \{x_i | x_i \in X_t \wedge y_i = c\}$, such that it maximizes the impurity gain when partitioning X_t w.r.t. the similarity with p^-. Formally, if $X_{t,l} = \{x_i \in X_t | s(x_i, p^-) \geq \beta_t\}$, $X_{t,r} = \{x_i \in X_t | s(x_i, p^-) < \beta_t\}$ and $Y_{t,l}, Y_{t,r}$ are the associated class labels, respectively, then if $\delta_s(p^-, X_t, Y_t)$, is the Information Gain calculated as in [8] maximizing a task-dependent measure like Entropy or Gini w.r.t. the similarities between p^- and X_t (instead of w.r.t. the features $\mathbb{R}^m$ of X_t), it does not exist another instance $\hat{p}^-$ such that $\delta_s(\hat{p}^-, X_t, Y_t) > \delta_s(p^-, X_t, Y_t)$. Furthermore, besides discriminative pivots, for each iteration v, PivotTree also identifies *representative* pivots. A representative pivot is an instance of class c, i.e., $p^+ \in \{x_i | x_i \in X_t \wedge y_i = c\}$ that maximizes the similarity with all the other instances described by the same node and belonging to the same class, i.e., $p^+ = \arg\max_{p' \in X_t^{(c)}} \sum_{x_i \neq p' \in X_t^{(c)}} s(x_i, p')$.

In Algorithm 1, we illustrate the pseudo-code for training a PivotTree. Given the dataset and labels $\langle X, Y \rangle$, the similarity function s, the maximum tree depth *maxdepth*, it returns the set P set of selected pivots, and the trained decision tree T (line 5). After initializing the tree and pivots (line 1), PivotTree induces a similarity matrix S between all pairs of instances in X (line 2). Then, the recursive procedure PTR is started (line 3). If the current depth of the tree DEPTH(T) is lower than the maximum tree depth *maxdepth* (line 6), then for each class, the most discriminative and most representative pivots are

Algorithm 1 PivotTree(X, Y)

Input: $\langle X, Y \rangle$ data and labels, s similarity function, *maxdepth* maximum tree depth
Output: P set of pivots, T learned tree

1: $T \leftarrow \emptyset$; $P \leftarrow \emptyset$; ▷ Variables initialization
2: $S \leftarrow \langle s(x_i, x_j) \rangle \; \forall x_i, x_j \in X \times X$ ▷ Calculate similarity matrix
3: $P, T \leftarrow \text{PTR}(X, Y, T, P, S)$; ▷ Start of recursive procedure
4: **return** P, T

5: **function** $\text{PTR}(X, Y, T, P, S)$
6: **if** $\text{DEPTH}(T) \leq maxdepth$ **then**
7: **for** $c \in C$ **do**
8: $p^- \leftarrow \arg\min_{x' \in X^{(c)}} \delta_s(x', X, Y)$; ▷ Get discriminative pivot
9: $p^+ \leftarrow \arg\max_{x \in X^{(c)}} \sum_{x \neq x' \in X^{(c)}} s(x, x')$; ▷ Get representative pivot
10: $P \leftarrow P \cup \{p^-, p^+\}$; ▷ Add pivots to result set
11: $X_l, X_r, Y_l, Y_r \leftarrow \text{SplitData}(X, Y, P)$; ▷ Split data w.r.t. P
12: $P_l, T_l \leftarrow \text{PTR}(X_l, Y_l, T, P, S)$ ▷ Recourse on left child
13: $P_r, T_r \leftarrow \text{PTR}(X_r, Y_r, T, P, S)$ ▷ Recourse on right child
14: $T \leftarrow \text{AddSplitToTree}(T, T_l, T_r)$; ▷ Add split to tree
15: **return** P, T; ▷ Return current pivots and tree
16: **else**
17: $p^+ \leftarrow \arg\max_{x \in X^{(c)}} \sum_{x \neq x' \in X^{(c)}} s(x, x')$; ▷ Get representative pivot
18: $P \leftarrow P \cup \{p^+\}$; ▷ Add pivots to result set
19: **return** $P, \text{MakeLeaf}(T)$; ▷ Return current pivots and leaf

selected and added to the result set P (lines 7–10)[5]. We notice that, since the similarity matrix S is calculated at the beginning, the pairwise similarities to select the most discriminative and representative pivots are available without performing any calculus. The set P of discriminative and representative pivots is then used to select the best split to partition the data with the SplitData function, again maximizing the Information Gain w.r.t. the similarities w.r.t. the pivots in P (line 11). We highlight that, by construction, SplitData selects a discriminative pivot. However, we keep these aspects separated as it is possible to run PivotTree relying only on representative pivots. After that, PivotTree recourses on the left and right subsets X_l, Y_l and X_r, Y_r and composes the tree returned (lines 12–14). On the other hand, if the maximum depth (line 16) or other stopping conditions are met, then the current pivots, augmented with the descriptive pivots of the records in the leaf, and a leaf itself (lines 17–19), are returned. Thus, the complexity of the PivotTree is theoretically bounded by the calculus of the similarity matrix S.

Furthermore, besides being used as a pivot selector method (π), we underline that PivotTree can be employed as a standalone predictive model by combin-

[5] To ease the computational burden, and similarly to other implementations, e.g., `scikit-learn`, we select only a subset of splits is evaluated.

Table 1: Datasets info: tr training and ts test size, m nbr. features, c nbr. classes.

	Image			**Tabular**										**Text**					**Time Series**				
name	cars	cifar	mnist	breast	compas	diva	german	heloc	house	iris	page	sonar	vertebral	wine	imdb	lyrics	news	vicuna	tgpt	devices	ecg	gun	wafer	worms
tr	9k	6k	6k	.4k	5k	9k	.8k	8k	16k	.2k	4k	.2k	.4k	5k	25k	26k	12k	7k	3k	9k	.6k	50	1k	.2k
ts	9k	1k	1k	.2k	2k	4k	.4k	4k	7k	45	2k	63	93	2k	25k	11k	8k	2k	.8k	8k	5k	.2k	7k	77
m	768	128	256	30	17	330	61	23	16	4	9	60	6	11	768	768	768	768	768	96	140	150	152	900
c	196	10	10	2	3	2	2	2	2	3	2	2	2	7	2	2	3	20	2	7	5	2	2	2

ing the encoding in the similarity space and the tree induction f. In this case, we do not need to train additional interpretable models, as both pivot selection and case-based prediction are already integrated into the model.

Data Agnosticism. By design, PivotTree is a data-agnostic model that leverages the concept of similarity to conduct both selection and prediction tasks simultaneously. While some data types, e.g., relational data, are more amenable than others, e.g., images or text, to similarity computation, with our contribution, we aim to address all data types as one. By decoupling similarity computation and object representation, PivotTree can be applied to any data type supporting a mapping to $\mathbb{R}^m$, i.e., text through language model, images through vision models, graphs through graph models, etc. In the following experimentation, besides tabular data, we focus on time series, images, and text.

4 Experiments

In this section, we evaluate the performance of PivotTree, which we implemented in Python[6], on different datasets with different modalities, and against a wide array of competitors. Our objective is to demonstrate that PivotTree is as accurate as state-of-the-art pivot selection methods, while being simpler. With PTS, we indicate PivotTree used as Selector, while with PTC, we refer to PivotTree directly used as Classification model.

Baselines and Competitors. We compare PivotTree with the following baselines and state-of-the-art similarity-based approaches for pivot selection (π):

- RND: randomly selects instances from the training set to be used as pivots;
- RNC: same as RND, but instances are sampled separately from each class;
- KMS: runs kMeans [40] and adopts the centroids as pivots;
- KMD: runs kMedoids [40] and adopts the medoids as pivots;
- EBL: selects pivots according to the ε-ball algorithm[7] [6].

Regarding model selection, we performed grid searches over the hyper parameter space, selecting the best-performing model on a validation set. On RND, RNC, and KMS, the number of pivots $|P|$ is selected within a grid on $|P| \in [2, 32]$.

[6] https://github.com/msetzu/pivottree

[7] https://docs.seldon.io/projects/alibi/en/latest/methods/ProtoSelect.html.

On EBL, the grid search for ε is performed on an interval between the 2^{nd} and the 40^{th} quantile of the empirical similarity distribution, as suggested in [6]. Regarding the interpretable predictive models (f) to be used on the selected prototypes, we rely on kNN and Decision Tree as implemented by the `sklearn` Python library. For PivotTree, both used as selector or predictor, i.e., PTS or PTC, the best *maxdepth* is searched in an interval $[2, 4]$. Obviously, a deeper PivotTree yields the selection of a larger number of pivots. Finally, to guarantee interpretability for the predictive models, we fix the hyper parameters as follows. Maximum depth equals four for Decision Trees [3], and the maximum number of neighbors for kNN equals to five [19]. As further baselines, we also compare PivotTree with kNN and DT directly trained on the original feature space while preserving hyper parameters.

Evaluation Measures. We evaluated the effectiveness of the selected pivots by measuring the F1-score of the predictive models relying on the different sets of pivots[8]. In line with the literature [7], as proxy of interpretability, we evaluated the *complexity* in terms of k, the number of selected pivots. Note that k can either be user-given, or optimized w.r.t. a given validation set. We experiment in both settings. Finally, to account for differences in datasets, and ease comparison, we turn complexity into *simplicity* as $1 - \frac{k}{|X|}$.

Datasets. In order to show the effectiveness of our proposal for different data types, we experimented with 11 tabular datasets, 5 time series datasets, 3 image datasets, and 5 text datasets. Table 1 reports some dataset details[9]. For tabular datasets, in order to perform a direct distance comparison between instances, we leave unvaried numeric and ordinal features while we one-hot encode categorical ones. We discard instances presenting missing values for one or more features. The datasets are then normalized with a z-score normalization by removing the mean and scaling to unit variance. Time series datasets are left unchanged as they are already preprocessed and normalized. For textual datasets, we first embed the input text with the `all-mpnet-base-v2` sentence transformer model[10], which yields $L2$-normalized 768-dimensional dense vector with magnitude 1. Finally, for image datasets, we embed each dataset with pretained and fine-tuned vision models. Further details are provided in the project repository. On the basis of these encodings, the similarity s is based on the Euclidean distance. While text embeddings usually rely on cosine similarity, in [29] it is shown that under unit normalization, the two are directly proportional and thus order-preserving.

Tabular datasets are divided into 70% training and 30% testing, while non tabular data sets come with their own split into training and test set. During model selection, a further split is performed, allocating for each development set 80% of the instances for training and 20% for validation. Thus, for each pivot selection method and classification method of each dataset, we perform a hold-out model-selection procedure, i.e., we find the best-performing hyper parameters

[8] For multi-class datasets we calculate the metric for each label and report the unweighted mean.

[9] The links to the various repositories and detailed preprocessing steps for the different datasets are available on the project repository.

[10] `https://huggingface.co/sentence-transformers/all-mpnet-base-v2`

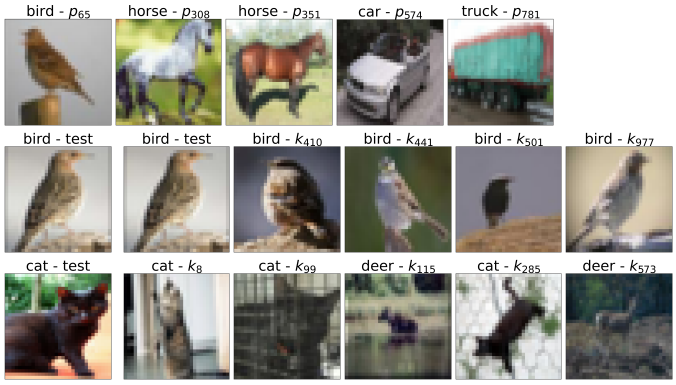


Fig. 2: PivotTree prediction and explanation on `cifar`. Top: selected pivots. Center and bottom: two classification examples. On the left a test instance; in the center, the five nearest neighbors of the selected pivots; and on the right a case-based decision rule.

configuration on the validation set and use it in the model-assessment phase, training on the whole training set and considering the resulting performances on the test set for final assessment.

Qualitative Results. In the following, we illustrate some qualitative examples on different data types to show the usability of PivotTree at prediction and explanation time with DT and kNN with the same set of pivots. PivotTree selects a set of pivots, which are then the training set for either a kNN or a DT. In the latter case, the data is first encoded in a pivot-instance similarity matrix. In Figure 2, we report two prediction and explanation examples on `cifar`. The top rows illustrated the pivots selected by PivotTree. The central and bottom rows show two classification examples for the *bird* and *cat* test instances, both on the left of the respective rows. Next to the test instances, we display the five neighbors selected by a kNN on the pivots similarity space. We can notice that for the *bird* example, all the neighbors are indeed birds quite similar in color and shape to the test instance. On the other hand, for the *cat* example, there are also some deers among the neighbors that however are in the same palette as the test instance. Finally, the right column shows the decision rules obtained by training a DT in the similarity space derived by PivotTree. We notice that the bird is recognized thanks to its dissimilarity with the car pivot p_{574} and its similarity with the bird pivot p_{65}. On the other hand, the cat is classified due to its dissimilarity with p_{351} and p_{781}. Thus, similarly to humans, these kinds of models can also reason by exclusion, suggesting their applicability also in the context of few-shot learning.

Similarly, Figure 3 reports two examples from the `gun` and `ecg` datasets, classifying tracked hand movements as gun draws and holsterings or not, and heartbeats of five different types, respectively. For both cases, the test instance has a large set of peculiarly similar neighbors, each with minimum variations.

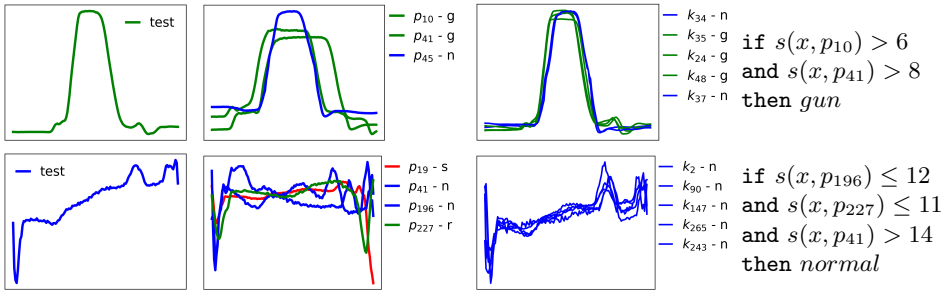


Fig. 3: Test time series (1^{st} column), pivots extracted by PivotTree (2^{nd} column), neighbors selected by kNN (3^{rd} column) and decision rule (4^{tg} column) on the `gun` (top) and `ecg` (bottom) datasets.

For the `gun` dataset, PivotTree has identified three pivots, two for the *not a gun* class, both characterized by short starting and ending movements, and interleaved by a long plateau. Here, the movement is sharp, but somewhat smooth, especially when gun is drawn, rather than holstered. The third pivot, associated to the *gun* class, is instead characterized by minimal motions, interleaved by a sharp draw, and a short plateau. The more pronounced movement closely resembles the test instance, but for a slight shift, and the instance is correctly classified by kNN as *gun*. The decision rule of the DT instead recognizes the *gun* class due to the similarity with p_{10} and p_{45}. For `ecg`, PivotTree identified four pivots, the test instance is very similar to p_{196} in the initial part and to p_{41} in the final part. The kNN classifier correctly retrieves neighbors with this shape and the test is correctly classified as *normal*. The DT instead distinguishes the normal class due to its limited similarity with p_{227} and high similarity with p_{41}.

Quantitative Results. Table 2 and Table 3 report the predictive model performance (F1-score) and complexity (number of pivots), respectively, per data modality and predictive model, i.e., DT and kNN. The bottom rows of the table report the average performance and standard deviations for all methods, and the rank of the pivot selection methods. The best and second-best performers per dataset among the pivot selection methods are in bold and italic, respectively. We can notice that when relying on the original data representation, i.e., when using directly DT or kNN on the training data, we have slightly better performance at the cost of losing the interpretability for non-tabular datasets. Focusing on predictive models relying on pivots, we notice that EBL has, on average, the highest F1-score (Table 2) immediately followed by PTS both for DT and kNN. We observe that the difference in the average of F1-score between EBL and PTS is only 0.1. All the other approaches follow them, with PTC being worse than EBL, thus indicating that PivotTree, in its current implementation, works better as a selector than as a classifier. On the other hand, concerning the complexity (Table 3), even though PTS is not minimizing the number of pivots

Table 2: Predictive model performance as F1-score.

	predictor	*Decision Tree*								*kNN*						
	selector	-	RND	RNC	KMS	KMD	EBL	PTS	PTC	-	RND	RNC	KMS	KMD	EBL	PTS
Img	cars	*.01*	**.02**	**.02**	**.02**	**.02**	**.02**	**.02**	.00	.86	.75	**.84**	.78	.75	*.80*	**.84**
	cifar	.75	*.40*	*.40*	*.40*	**.41**	**.41**	.39	.11	.87	*.87*	*.87*	**.88**	**.88**	**.88**	*.87*
	mnist	.68	*.44*	.41	**.53**	*.44*	.41	.37	.29	.97	*.96*	*.96*	*.96*	*.96*	**.97**	*.96*
Tabular	breast	.95	*.94*	.93	.93	*.94*	**.95**	*.94*	**.95**	.96	*.95*	*.95*	*.95*	**.96**	**.96**	*.95*
	compas	.50	.47	*.48*	.46	*.48*	*.48*	.52	**.49**	.46	*.47*	**.48**	**.48**	**.48**	.46	*.47*
	german	.58	*.54*	.53	.48	.48	**.61**	.50	.48	.67	.59	.59	.59	.59	**.65**	*.60*
	heloc	.70	.66	.66	.65	.66	*.67*	**.68**	.66	.67	*.66*	*.66*	**.67**	**.67**	**.67**	*.66*
	house	.80	.72	.72	.73	.73	**.78**	.76	*.77*	.83	*.80*	.79	*.80*	*.80*	**.82**	*.80*
	iris	1.0	*.94*	.91	**.96**	**.96**	.90	.92	.92	1.0	*.99*	**1.0**	**1.0**	**1.0**	**1.0**	.98
	page	.88	.84	.86	*.87*	.85	**.88**	.84	*.87*	.90	.88	*.89*	**.90**	*.89*	**.90**	.88
	diva	.79	.60	.59	.58	.56	**.64**	**.64**	*.63*	.76	.72	.72	.70	.71	**.75**	*.73*
	sonar	.74	.70	*.73*	.72	.71	**.77**	*.73*	.59	.94	.82	*.84*	.83	.81	**.89**	*.84*
	vert.	.72	.68	*.69*	.66	*.69*	.65	**.71**	.68	.73	.69	.69	*.73*	*.76*	.74	**.78**
	wine	.20	*.19*	**.20**	*.19*	**.20**	**.20**	**.20**	.18	.37	*.35*	*.35*	*.35*	**.36**	*.35*	*.35*
Text	imdb	.70	.73	.72	.75	.74	**.79**	*.78*	*.78*	.78	.78	.77	.79	*.80*	.79	**.82**
	lyrics	.66	*.69*	*.69*	.68	.68	**.70**	**.70**	**.70**	.71	*.70*	*.70*	*.70*	*.70*	**.71**	**.71**
	news	.12	.16	.16	*.19*	.18	.16	**.24**	.01	.69	.55	.50	.62	.60	**.66**	*.65*
	tgpt	.84	.80	.80	.80	*.81*	**.84**	.79	**.84**	.92	.88	.88	**.90**	**.90**	*.89*	*.89*
	vicuna	.63	.57	.55	.55	.59	**.64**	*.63*	.59	.68	.69	.69	.67	.71	**.73**	*.72*
TimeS	devices	.25	.33	.32	.34	.34	*.39*	**.42**	.34	.49	.47	.46	.48	.48	*.49*	**.52**
	worms	.54	.54	.53	**.57**	*.56*	*.56*	*.56*	*.56*	.60	.58	*.61*	.58	.58	*.61*	**.70**
	ecg	.52	.50	*.51*	.50	.50	**.53**	*.51*	*.51*	.57	.54	.54	*.55*	*.55*	**.56**	**.56**
	gun	.80	*.77*	*.77*	.76	**.78**	*.77*	.71	.74	.91	.87	.87	**.89**	**.89**	*.88*	.84
	wafer	.90	.93	.93	*.94*	*.94*	**.95**	.92	.93	.99	*.97*	**.98**	**.98**	**.98**	**.98**	**.98**
	avg	.64	*.59*	*.59*	*.59*	*.59*	**.61**	.60	.57	.76	.73	.73	.74	.74	**.76**	*.75*
	std	.26	*.25*	*.25*	*.25*	*.25*	*.25*	**.24**	.29	.18	*.18*	*.18*	*.18*	*.18*	*.18*	**.17**
	rank		4.8	4.8	4.21	3.9	**2.5**	*3.5*	4.3		4.9	4.4	3.6	3.15	**2.2**	*2.9*

selected compared to other methods such as KMS, it still requires less than half of the pivots used by EBL to guarantee comparable performance.

The non-parametric Friedman test compares the average ranks of the various methods over multiple datasets w.r.t. an evaluation measure, in our case, F1-score and complexity. The null hypothesis that all methods are equivalent is rejected ($p < 0.001$) for all the experiments reported in the various tables. The comparison of the ranks of all methods against each other can be visually represented as shown by the critical difference plots in Figure 4: lower rank values indicate better models, i.e., best ranks on the right (see [16] for details). In Figure 4, methods statistically equivalent according to a post-hoc Nemenyi test are connected by black lines. We notice that regardless of the classification model f used, EBL and PTS are tied w.r.t F1-score, while PTS is significantly less complex and untied w.r.t. the number of pivots selected.

In summary, PTS is the best pivot selector, achieving high predictive performance with a smaller number of pivots. Such a result is best appreciated in Figure 5, where we show the mean and standard deviation of the F1-score and the simplicity of pivot selection methods. Besides, Figure 5 also highlights the lowest variability of PTS w.r.t EBL in terms of simplicity.

We repeated the experiments in a constrained setting[11] wherein pivot selection was limited to a maximum of 20 pivots (Table 4 and Figure 6). While the

[11] `cars` has not been used as it contains 196 classes, and all the methods would have failed.

Table 3: Predictive model complexity as number of pivots used.

	predictor	*Decision Tree*								*kNN*						
	selector	-	RND	RNC	KMS	KMD	EBL	PTS	PTC	-	RND	RNC	KMS	KMD	EBL	PTS
Img	cars	-	10	196	*6*	*6*	64	778	**4**	-	*32*	196	*32*	**28**	64	974
	cifar	-	32	**10**	18	28	220	118	*12*	-	32	*20*	22	30	**18**	42
	mnist	-	*4*	10	*4*	*4*	261	**2**	12	-	32	**20**	*30*	*30*	133	73
Tabular	breast	-	32	24	*20*	28	88	39	**6**	-	26	28	*12*	**6**	99	20
	compas	-	32	32	30	18	70	**9**	*10*	-	*18*	*18*	30	32	581	**7**
	german	-	26	32	24	24	60	*22*	**10**	-	**32**	**32**	**32**	**32**	*72*	**32**
	heloc	-	28	32	*24*	32	880	**9**	**9**	-	32	32	*18*	22	378	**9**
	house	-	32	32	16	20	2k	**6**	*13*	-	32	**26**	*28*	32	1k	30
	iris	-	28	32	28	*4*	69	16	**3**	-	28	28	32	28	*20*	**10**
	page	-	32	32	*22*	24	105	69	**10**	-	30	32	**6**	*20*	112	**6**
	diva	-	*30*	32	32	*30*	528	83	**13**	-	32	*28*	30	30	311	**13**
	sonar	-	32	32	22	22	26	*21*	**4**	-	32	*20*	22	24	21	**6**
	vert.	-	30	32	*28*	**8**	61	53	**8**	-	32	10	18	*8*	21	**3**
	wine	-	28	28	*22*	*22*	150	158	**14**	-	*24*	28	**22**	32	32	121
Text	imdb	-	32	32	*10*	18	531	26	**8**	-	32	32	**8**	30	980	*26*
	lyrics	-	30	32	30	*24*	5k	*24*	**2**	-	32	32	*30*	**28**	156	99
	news	-	30	*20*	*20*	22	215	106	**13**	-	32	**20**	*32*	*32*	215	844
	tgpt	-	26	32	*14*	26	247	18	**12**	-	*32*	*32*	*32*	**28**	187	68
	vicuna	-	32	32	32	*14*	107	40	**12**	-	*32*	*32*	*32*	*32*	540	**30**
TimeS	devices	-	32	28	24	**8**	136	408	*12*	-	32	*28*	**26**	30	896	89
	worms	-	32	30	*12*	16	107	25	**6**	-	**8**	24	26	*14*	32	20
	ecg	-	28	28	16	24	43	*14*	**3**	-	30	**24**	30	*26*	96	37
	gun	-	28	20	*8*	30	15	**2**	**2**	-	20	24	8	20	*5*	**4**
	wafer	-	26	28	*12*	24	43	13	**3**	-	**30**	*32*	*32*	**30**	43	45
	avg	-	28	35	*20*	*20*	523	86	**8**	-	29	33	**24**	*26*	259	109
	std	-	*7*	35	8	8	1k	170	**4**	-	**6**	35	8	*7*	338	249
	rank		4.8	4.9	*3.*	3.3	6.6	3.9	**1.4**		3.7	*3.1*	**2.9**	**2.9**	5.1	3.3

average performance remains more or less unchanged, we notice that PTS is the best performer among the various competitors when DT is used as a classifier. On the contrary, PTC worsens its ranking. In other terms, PivotTree excels in different settings according to the number k of pivots extracted: when k is small, a Decision Tree is best; and when k is large, then kNN is best.

Sensitivity Analysis. Figure 7 reports a sensitivity analysis on PivotTree used as pivot selector (PTS). In particular, we observe the average F1-score among all datasets with error bars indicating the standard deviations when varying the maximum number of pivots in ranges from 10 to 20, from 20 to 30, etc. Two lines are reported to differentiate the performance between datasets with 2 or 3 classes, i.e., $c \in [2, 3]$, versus datasets with 5 to 10 classes, i.e., $c \in [5, 10]$. We leave as a future study a sensitivity analysis of datasets with more than 10 classes. The results show that, both for DT and kNN, for datasets with few classes, the performance is stable independently of the number of pivots selected. Thus, extracting a limited number of highly discriminative and representative pivots can guarantee high performance and high simplicity. On the other hand, for datasets with more than five classes, the results are less stable, and we observe an increase in performance, especially when using kNN, as the DT we relied on is limited by the maximum depth of four, thus practically being limited by its depth and not exploiting all the possible pivots. As a consequence, for datasets with a high number of classes, the tuning of the number of pivots k extracted

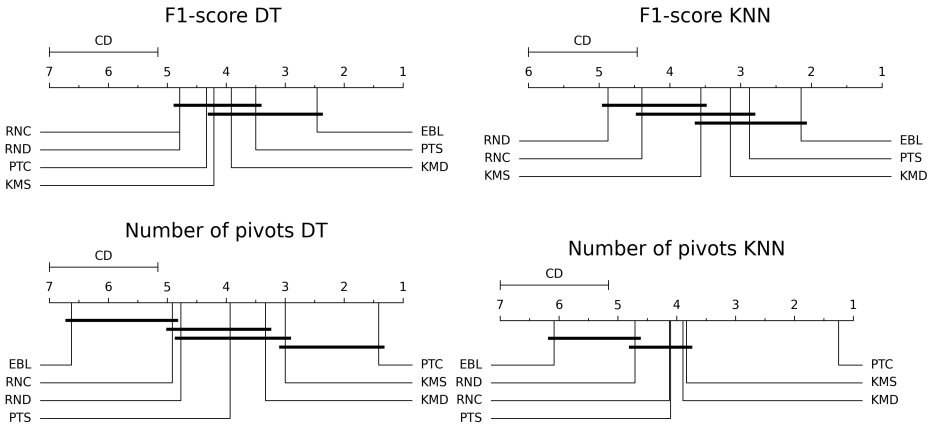


Fig. 4: Comparison of model's rank w.r.t. F1-score and complexity against each other with the Nemenyi test. Groups of classifiers that are not significantly different at 95% significance level are connected. Best ranks on the right.

Table 4: Model performance as F1-score with models limited to 20 pivots.

	predictor	*Decision Tree*								*kNN*						
	selector	-	RND	RNC	KMS	KMD	EBL	PTS	PTC	-	RND	RNC	KMS	KMD	EBL	PTS
Img	cifar	.75	.39	.40	.40	*.41*	*.41*	**.42**	.11	.87	.86	*.87*	**.88**	**.88**	**.88**	.80
	mnist	.68	*.44*	.41	**.53**	*.44*	.42	.37	.29	.97	**.96**	**.96**	**.96**	**.96**	**.96**	*.95*
Tabular	breast	.95	.93	.94	.93	.94	.94	**.96**	*.95*	.96	.95	.95	.95	*.96*	**.97**	.95
	compas	.50	.46	.48	.46	.48	.47	**.52**	*.49*	.46	.47	*.48*	*.48*	.47	**.49**	.47
	german	.58	.52	*.53*	**.56**	.49	.44	.50	.48	.67	.58	.58	*.59*	.58	**.60**	.58
	heloc	.70	.65	.65	.65	.65	.65	**.68**	*.66*	.67	*.66*	.65	**.67**	**.67**	**.67**	*.66*
	iris	1.0	.94	.92	**.98**	*.96*	.95	.95	.92	1.0	.95	**1.0**	**1.0**	**1.0**	**1.0**	*.98*
	page	.88	.83	.83	*.87*	*.87*	**.88**	.84	*.87*	.90	.88	.88	**.90**	*.89*	*.89*	.88
	diva	.79	*.60*	*.60*	.58	.55	.59	**.63**	**.63**	.76	*.72*	.71	.69	.70	.71	**.73**
	sonar	.74	.71	.70	.71	.71	**.74**	*.72*	.59	.94	.82	*.84*	.82	.83	**.89**	*.84*
	vert.	.72	.66	**.69**	.66	**.69**	.66	.66	*.68*	.73	.69	.69	.73	*.76*	.74	**.78**
	wine	.20	**.19**	**.19**	**.19**	**.19**	*.18*	**.19**	*.18*	.37	*.35*	*.35*	*.35*	*.35*	*.35*	**.36**
Text	imdb	.70	.72	.71	.75	.74	*.77*	**.78**	**.78**	.78	.76	.75	*.79*	*.79*	**.81**	**.81**
	lyrics	.66	*.68*	*.68*	*.68*	*.68*	**.70**	**.70**	**.70**	.71	.69	.69	*.70*	.68	**.71**	.68
	news	.12	.15	.16	**.19**	*.18*	.16	**.19**	.01	.69	.48	.50	**.58**	*.55*	**.58**	.40
	tgpt	.84	.79	.79	.80	*.82*	**.84**	.79	**.84**	.92	.87	.86	*.88*	*.88*	**.90**	*.88*
	vicuna	.63	.55	.55	.52	*.59*	.57	**.64**	*.59*	.68	.67	.67	.66	.69	**.72**	*.71*
TimeS	devices	.25	.32	.30	.30	.34	*.36*	**.38**	.34	.49	.46	.44	*.48*	**.49**	.46	.47
	worms	.54	.52	.52	*.57*	.56	**.61**	.56	.56	.60	.58	*.61*	.60	.58	.51	**.70**
	ecg	.52	.49	**.51**	*.50*	*.50*	.46	**.51**	**.51**	.57	.54	.54	*.55*	*.55*	*.55*	**.56**
	gun	.80	**.77**	**.77**	*.76*	**.77**	**.77**	.71	.74	.91	.87	.86	**.89**	**.89**	*.88*	.84
	wafer	.90	*.93*	*.93*	**.94**	*.93*	*.93*	.92	*.93*	.99	*.97*	**.98**	**.98**	**.98**	**.98**	**.98**
	avg	.66	.60	.60	**.62**	*.61*	*.61*	**.62**	.58	.76	.72	.72	*.73*	*.73*	**.74**	*.73*
	std	.23	**.22**	**.22**	**.22**	**.22**	*.23*	**.22**	.27	.18	*.19*	*.19*	**.18**	*.19*	*.19*	*.19*
	rank		5.0	4.6	4.0	*3.6*	3.7	**3.1**	3.9		4.8	4.1	*3.0*	3.1	**2.2**	3.5

with PivotTree should be carefully addressed, and it should consider a high number potentially limiting the final interpretability of the predictive model.

Although time complexity is not the primary focus of this paper, here we also report training runtime (in seconds). As example for small datasets, *breast* and

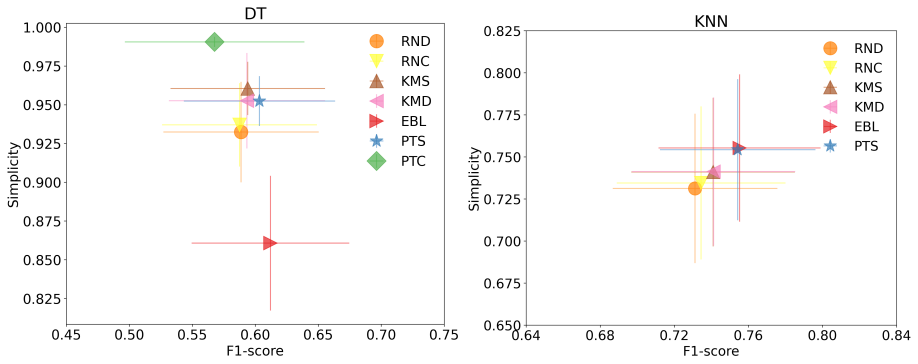


Fig. 5: Scatter plots for average F1-score and simplicity for pivot selection methods with error bars reporting 10% of the standard deviation.

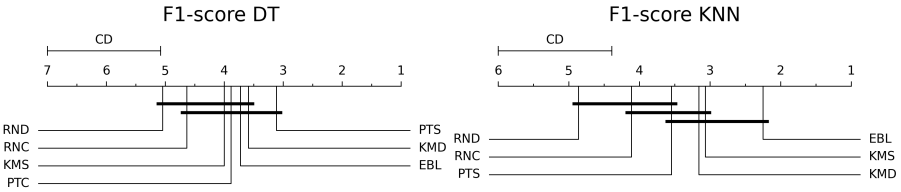


Fig. 6: Comparison of model's rank w.r.t. F1-score against each other with Nemenyi test. Classifiers that are not significantly different at 95% significance level are connected. Best ranks on the right. Models limited to 20 pivots.

ecg datasets present fitting runtimes respectively of 4.34s and 8.29s. In contrast, *tgpt* and *cifar* show higher training times of 24.91s and 60.70s. Larger datasets, both in terms of instances and dimensions, require longer training times, due to the need of finding pivots within a bigger pool. For example, *lyrics* requires 458.81s for training. In all cases mentioned, prediction times are relatively fast, with all predictions taking under 24.10s, which is the time needed to perform predictions for the *imdb* test set.

5 Conclusions

We have introduced PivotTree, an interpretable tree-based pivot selection model aimed at facilitating the training of effective interpretable case-based predictive models. In PivotTree, exemplary instances, named *pivots*, guide the construction of a similarity-based case-based model where explanations are a hierarchy of prototypical instances. By design, PivotTree is both a pivot selector and a prediction model, enabling, independently, both the extraction of relevant instances and the construction of an interpretable predictive model. PivotTree is a *data-agnostic* model, which can be seamlessly applied to various

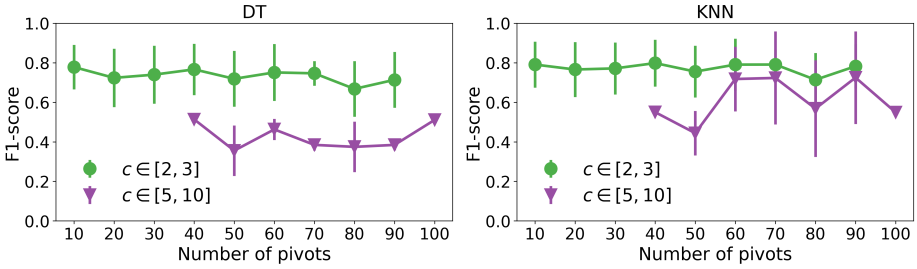


Fig. 7: F1-score varying the number of pivots w.r.t. bins of pivots for datasets with different number of classes.

data modalities, including tabular data, text, time series, and images. In a wide array of experiments, PivotTree has shown to be on par with state-of-the-art approaches while often retaining lower complexity and higher interpretability.

Given its inherent flexibility, PivotTree lends itself to several future improvements: different data encodings, e.g. TabPFN [23] or Rocket [15], may further improve instance representation, and thus similarity estimation; joint optimization of pivot selection and case-based reasoning, which is currently decoupled in pivot selection, and tree induction; use of more sophisticated case-based reasoning models; adaptation for other data types such as mobility trajectories [30], and evaluation of the privacy exposure lead by pivots [34]. Another avenue of research lies in integrating prior knowledge or human supervision into prototype learning, as human-machine collaboration could improve the classifier's accuracy and interpretability, as suggested and investigated in [33, 42]. Furthermore, future avenues of research also include assessing PivotTree's interpretability from a human-centric perspective, validating its performance through evaluation schema designed for prototype-based explanations, as described for images in [13,28], time-series in [33], and texts in [12,24]. As such, an extensive comparison of PivotTree's performance and explainability could be conducted against deep learning-based *representations* of the prototypes across different modalities, as well as through feature-based explainability techniques.

Acknowledgments. This work has been partially supported by the European Community Horizon 2020 programme under the funding schemes ERC-2018-ADG G.A. 834756 "XAI: Science and technology for the eXplanation of AI decision making" (https://xai-project.eu/), "INFRAIA-01-2018-2019 – Integrating Activities for Advanced Communities", G.A. 871042, "SoBigData++: European Integrated Infrastructure for Social Mining and Big Data Analytics" (http://www.sobigdata.eu), G.A. 101120763 TANGO (https://tango-horizon.eu/), 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 – Avviso n. 3264 del 28/12/2021, and M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI", and by the Italian Project Fondo Italiano per la Scienza FIS00001966 MIMOSA, by the Euro-

pean Union, Next Generation EU, within the PRIN 2022 framework project PIANO (Personalized Interventions Against Online Toxicity) under CUP B53D23013290006.

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

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I.J., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: NeurIPS. pp. 9525–9536 (2018)
2. Angelov, P.P., Soares, E.A.: Towards explainable deep neural networks (xdnn). Neural Networks **130**, 185–194 (2020)
3. Bertsimas, D., Dunn, J.: Optimal classification trees. MACH (2017)
4. Bichindaritz, I., Marling, C.: Case-based reasoning in the health sciences: What's next? Artif. Intell. Medicine **36**(2), 127–135 (2006)
5. Bien, J., Tibshirani, R.: Hierarchical clustering with prototypes via minimax linkage. Journal of the American Statistical Association **106**(495), 1075–1084 (2011)
6. Bien, J., Tibshirani, R.: Prototype selection for interpretable classification. The Annals of Applied Statistics pp. 2403–2424 (2011)
7. Bodria, F., Giannotti, F., et al.: Benchmarking and survey of explanation methods for black box models. DMKD **37**(5), 1719–1778 (2023)
8. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth (1984)
9. Chatzakou, D., Leontiadis, I., et al.: Detecting cyberbullying and cyberaggression in social media. ACM Trans. Web **13**(3), 17:1–17:51 (2019)
10. Chen, C., et al.: This looks like that: Deep learning for interpretable image recognition. In: NeurIPS. pp. 8928–8939 (2019)
11. Chui, M., Hall, B., Mayhew, H., Singla, A., Sukharevsky, A., by McKinsey, A.: The state of ai in 2022-and a half decade in review. Mc Kinsey (2022)
12. Das, A., et al.: Prototex: Explaining model decisions with prototype tensors. In: ACL (1). pp. 2986–2997. Association for Computational Linguistics (2022)
13. Davoodi, O., et al.: On the interpretability of part-prototype based classifiers: A human centric analysis. CoRR **abs/2310.06966** (2023)
14. De Fauw, J., et al.: Clinically applicable deep learning for diagnosis and referral in retinal disease. Nature medicine **24**(9), 1342–1350 (2018)
15. Dempster, A., et al.: ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. DMKD **34**(5), 1454–1495 (2020)
16. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. JMLR (2006)
17. Fix, E.: Discriminatory analysis: nonparametric discrimination, consistency properties, vol. 1. USAF school of Aviation Medicine (1985)
18. Frosst, N., Hinton, G.E.: Distilling a neural network into a soft decision tree. In: CEx@AI*IA. CEUR Workshop Proceedings, vol. 2071. CEUR-WS.org (2017)
19. Golding, A.R.: A review of case-based reasoning. AI Mag. **16**(2), 85–86 (1995)
20. Guidotti, R., Monreale, A., et al.: A survey of methods for explaining black box models. ACM CSUR **51**(5), 93:1–93:42 (2019)
21. Guidotti, R., Rossetti, G., et al.: Personalized market basket prediction with temporal annotated recurring sequences. IEEE TKDE **31**(11), 2151–2163 (2019)

22. Hase, P., Chen, C., Li, O., Rudin, C.: Interpretable image recognition with hierarchical prototypes. In: HCOMP. pp. 32–40. AAAI Press (2019)
23. Hollmann, N., Müller, S., Eggensperger, K., Hutter, F.: Tabpfn: A transformer that solves small tabular classification problems in a second. In: ICLR (2023)
24. Hong, D., Wang, T., Baek, S.: Protorynet-interpretable text classification via prototype trajectories. JMLR **24**(264), 1–39 (2023)
25. Jeyakumar, J.V., et al.: How can I explain this to you? an empirical study of deep neural network explanation methods. In: NeurIPS (2020)
26. Johnson-Laird, P.N.: Mental models and human reasoning. Proceedings of the National Academy of Sciences **107**(43), 18243–18250 (2010)
27. Kasirzadeh, A., Clifford, D.: Fairness and data protection impact assessments. In: AIES. pp. 146–153. ACM (2021)
28. Kim, S.S.Y., et al.: HIVE: evaluating the human interpretability of visual explanations. In: ECCV (12). LNCS, vol. 13672, pp. 280–298. Springer (2022)
29. Korenius, T., Laurikkala, J., Juhola, M.: On principal component analysis, cosine and euclidean measures in information retrieval. Inf. Sci. **177**(22) (2007)
30. Landi, C., et al.: Geolet: An interpretable model for trajectory classification. In: IDA. LNCS, vol. 13876, pp. 236–248. Springer (2023)
31. Li, W., et al.: A data-driven explainable case-based reasoning approach for financial risk detection. Quantitative Finance **22**(12), 2257–2274 (2022)
32. Lucas, B., Shifaz, A., et al.: Proximity forest: an effective and scalable distance-based classifier for time series. DMKD **33**(3), 607–635 (2019)
33. Ming, Y., et al.: Interpretable and steerable sequence learning via prototypes. In: KDD. pp. 903–913. ACM (2019)
34. Naretto, F., Monreale, A., Giannotti, F.: Evaluating the privacy exposure of interpretable global explainers. In: CogMI. pp. 13–19. IEEE (2022)
35. Nauta, M., van Bree, R., Seifert, C.: Neural prototype trees for interpretable fine-grained image recognition. In: CVPR (2021)
36. Nguyen, G., et al.: The effectiveness of feature attribution methods and its correlation with automatic evaluation scores. In: NeurIPS. pp. 26422–26436 (2021)
37. Pekalska, E., Duin, R.P.W.: The Dissimilarity Representation for Pattern Recognition, Series in MPAI, vol. 64. WorldScientific (2005)
38. 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)
39. Spelke, E.S.: What babies know: Core knowledge and composition, Volume 1. Oxford University Press (2022)
40. Tan, P.N., Steinbach, M., Kumar, V.: Data mining introduction. People's Posts and Telecommunications Publishing House, Beijing (2006)
41. Waa, J.V.D., et al.: Evaluating XAI: A comparison of rule-based and example-based explanations. Artificial Intelligence **291** (2021)
42. Xie, J., et al.: Prototype learning for medical time series classification via human-machine collaboration. Sensors **24**(8), 2655 (2024)
43. Yang, G., et al.: Unbox the black-box for the medical explainable AI via multi-modal and multi-centre data fusion. Inf. Fusion **77**, 29–52 (2022)
44. Zhang, Q., Yang, Y., Ma, H., Wu, Y.N.: Interpreting CNNs via Decision Trees. In: CVPR. pp. 6261–6270. Computer Vision Foundation / IEEE (2019)
45. Zhang, X., Jiang, S.: A splitting criteria based on similarity in decision tree learning. J. Softw. **7**(8), 1775–1782 (2012)