Achieving Complete Coverage with Hypercube-Based Symbolic Knowledge-Extraction Techniques

Federico Sabbatini[1](✉) and Roberta Calegari[2]

[1] Department of Pure and Applied Sciences (DiSPeA), University of Urbino "Carlo Bo", Urbino, Italy
f.sabbatini1@campus.uniurb.it

[2] Department of Computer Science and Engineering (DISI), University of Bologna, Bologna, Italy
roberta.calegari@unibo.it

Abstract. Symbolic knowledge-extraction (SKE) techniques are currently employed for various purposes, particularly addressing the challenge of explaining opaque models by generating human-understandable explanations. The existing literature encompasses a diverse range of techniques, each relying on specific theoretical assumptions and possessing its own advantages and disadvantages. Amongst the available choices, hypercube-based SKE techniques are notable for their adaptability and versatility. However, they may suffer from limited completeness when utilised for making predictions. This research aims to augment the predictive capabilities of hypercube-based SKE techniques, striving for a completeness rate of 100%. Furthermore, the study includes experiments that assess the effectiveness of the proposed enhancements.

Keywords: Explainable artificial intelligence · Symbolic knowledge extraction · PSyKE

1 Introduction

Ensuring the explainability of predictions made by machine learning (ML) models is crucial, especially in critical domains where the outcomes significantly impact human well-being, such as health, wealth, and safety. To address the opacity of ML predictors, the explainable artificial intelligence (XAI) community proposes two primary approaches [8]: *(i)* utilizing inherently human-interpretable models, such as decision trees with limited depth [13]; or *(ii)* employing symbolic knowledge-extraction (SKE) techniques to extract post-hoc explanations from trained opaque models [12]. This paper focuses on SKE techniques.

Over the past few decades, numerous SKE algorithms have been proposed in the literature. While these techniques exhibit diversity, certain common characteristics can be identified amongst the most widely adopted approaches. For example, G-REX [11], Trepan [4] and Cart [2] offer human-understandable knowledge in the form of decision trees. However, it should be noted that Trepan is only applicable to binary input feature classification tasks. On the other hand, G-REX and Cart can be utilised also for regression tasks and accept inputs that are either discrete or continuous.

SKE techniques often exhibit a recurring pattern of employing hypercubic[1] partitioning of the input feature space [22,24]. This approach aims to generate interpretable predictions. The identification of specific regions within the input feature space, characterised by interval inclusion constraints – typically one constraint per input feature –, forms the basis of these methods. Consequently, the outputs of these techniques can be easily understood by human users. Each hypercubic region that is identified is associated with a comprehensible output value, which could be a class label, a constant value, or a linear combination of the input features.

SKE algorithms based on hypercubes can be executed using either a top-down or a bottom-up workflow [15]. The bottom-up approach is particularly susceptible to non-exhaustivity issues, referring to the potential inability to generate predictions for instances belonging to certain subregions of the input space. Generally, the bottom-up strategy involves iteratively expanding the hypercubes created in the input feature space, one cube and one dimension at a time. The cubes are initially defined as multidimensional points, and achieving convergence (i.e., complete coverage of the input feature space) may require a significant number of iterations, especially for data sets with numerous input features. Consequently, the presence of non-exhaustivity in bottom-up hypercube-based SKE techniques depends on the complexity of the data set being analyzed. An example of an SKE algorithm that suffers from this drawback is Iter [9].

Due to the time-consuming process of iterative hypercube expansion, bottom-up algorithms like Iter may terminate after a specific number of user-defined iterations, even if they have not yet achieved convergence. This means that certain subregions of the input space remain uncovered by the identified hypercubes, resulting in the inability to predict instances belonging to these uncovered subregions. As a result, the completeness of SKE becomes a crucial factor to consider when evaluating the quality of a technique [7].

The significance of completeness in assessing the quality of knowledge extracted through SKE methods is also emphasised in [18,19], where two metrics are introduced to evaluate knowledge quality. These metrics utilise other indices commonly employed to assess the quality of SKE techniques, namely correctness and compactness [7]. Correctness measures the ability of the SKE technique to replicate the predictions of the underlying opaque predictor. On the other hand, compactness is a measure of human readability, as knowledge with a smaller dimension is more understandable for end users compared to knowledge with a large size.

[1] We use the term "hypercube" also for referring to actual hyperrectangles, as commonly made in the literature [9, for instance].

In this paper, we address the significance of obtaining comprehensive interpretable models by introducing a set of vicinity-based extensions for hypercube-based SKE methods. These extensions aim to provide human-interpretable predictions even for instances that do not belong to any identified hypercube. The proposed extension is not limited to a specific SKE algorithm or task. Therefore, it can be applied to any type of SKE technique that relies on hypercubic partitioning of the input feature space, regardless of whether it involves classification or regression tasks with categorical, discrete, or continuous outputs.

The paper is organised as follows: Section 2 resumes background notions on SKE, Sect. 3 describes the proposed extensions, and Sect. 4 shows the effectiveness of our proposal via several experiments on real world data sets. Finally, Sect. 5 summarises our conclusions.

2 Symbolic Knowledge Extraction

Knowledge-extraction mechanisms usually aim to reverse-engineer an opaque model in order to understand the rationale behind the output predictions it provides [10]. SKE algorithms may be categorised along several dimensions, including (but not limited to) the expressive power of the extracted knowledge (e.g., propositional, fuzzy, oblique rules) and the *translucency* extent of the technique [1]. According to the translucency dimension, SKE methods may be classified as *decompositional* or *pedagogical*.

A group of pedagogical SKE techniques [9,14,20,25] rely on hypercubic partitioning of the input feature space to establish input/output relationships between queries made to opaque models and their corresponding output predictions. These techniques fall into a category that typically generates human-interpretable outcomes in the form of propositional rules presented as a list or tree structure. In the following, we delve into the specifics of these hypercube-based SKE methods to elucidate the potential causes of their non-exhaustivity. Additionally, we highlight the advantages of employing the presented extension to achieve complete coverage, addressing the limitations of these techniques.

2.1 Iter

Iter [9] is a bottom-up SKE technique originally designed for regression tasks described by continuous input features. It is based on the creation of a user-defined number of small hypercubes randomly placed inside the input feature space (i.e., multidimensional points) and on the iterative expansion of these cubes until the whole input space is covered (i.e., convergence) or it is not possible to further expand them. Hypercubes are always non-overlapping, to avoid ambiguity in the prediction phase.

The expansion step of Iter may terminate without reaching convergence after a user-defined number of iterations. In this case some portions of the input feature space remain unassociated with the found hypercubes and the resulting interpretable model will be unable to provide predictions for all instances belonging to these unassociated regions.

The non-exhaustivity of Iter is thus due to the slow convergence of its expansion phase, given that at each iteration only one cube is expanded along a single dimension and such expansion is generally represented by a small user-defined amount of input space.

2.2 GridEx and GridREx

GridEx [25] is a top-down algorithm to perform knowledge extraction from any kind of opaque predictor. It has been designed to overcome the non-exhaustivity issues deriving from the usage of Iter. To do so, GridEx recursively splits the input feature space in a set of symmetric, disjoint partitions according to some criteria. In particular, GridEx identify 3 classes of input space regions: negligible if there are no training samples belonging to them; permanent if they contain training samples and the associated predictive error is below the user-defined threshold; and eligible if they contain samples and the corresponding error is above the threshold. From a workflow standpoint, negligible regions are discarded, permanent regions are converted into human-readable rules and eligible regions are further split during the recursive phase of the algorithm.

A consequence of the splitting strategy adopted by GridEx is the possibility to produce a non-exhaustive input space partitioning when applied to data sets having sparse data points, due to the discarded negligible regions. In this case GridEx cannot predict instances falling inside these input space portions.

GridREx [14] is an extension of GridEx aimed at achieving better predictive performance. Since GridEx associates constant output values to the identified hypercubes, it introduces an undesired discretisation impinging the predictive performance of the interpretable model. GridREx overcomes this drawback by substituting the constant output of each hypercube with a regression law expressing a linear combination of the input features. However, it shares with GridEx the same splitting strategy and thus the same issues related to the non-exhaustivity due to negligible regions.

2.3 CReEPy

CReEPy [20] is a top-down SKE technique producing a hypercubic partitioning of the input feature space organised as a binary tree. It is based on an underlying explainable clustering algorithm that may be selected by users, together with the corresponding parameters. Available clustering techniques are ExACT [16] and CREAM [17].

The tree structure produced by this algorithm is created by recursively splitting input space regions into two subregions: a hypercube and a difference cube, obtained by subtracting the hypercubic subregion from the starting region. As a consequence, each node of the tree is associated with a constraint describing a hypercube. The two child nodes are then associated with inclusion in/exclusion from that hypercube.

Since CReePy produces human-intelligible rules by traversing the binary tree, it is always possible to provide an output prediction for a given query and therefore it is a complete technique by design.

3 Vicinity-Based Extensions to Achieve Complete Coverage

In order to address the completeness of the interpretable models obtained through SKE techniques, we have developed an extension that can be applied to any SKE algorithm utilizing hypercubic input space partitioning for predictions. The problem of drawing predictions for uncovered queries has already been investigated in the literature [5, 27]. In particular, [5] emphasises the need for a more sophisticated strategy than majority-based assignments and proposes an alternative method based on rule stretching. On the other hand, [27] exploits Euclidean distance to assign a point to the nearest hypercube, selected amongst a set of possibly *nested* hypercubes.

It is important to note that our proposed extension does not impact the prediction phase for data points that fall within the identified hypercubes during knowledge extraction. These instances can be predicted as intended by the design of the SKE algorithm. Instead, our focus is solely on predicting instances that are included in regions not covered by any hypercube, referred to as *uncovered instances* hereafter.

The core concept behind this extension is to assign each uncovered instance to an existing hypercube based on a vicinity criterion. This approach ensures that the readability extent of the interpretable model obtained through SKE remains unchanged, as it is directly related to the size of knowledge represented by the number of identified hypercubes, which remains unaltered with our extension. Additionally, employing a vicinity criterion enables the assignment of outliers to the closest hypercube. Experimental tests have shown that the proposed extension, which in the following we refer to as *brute prediction*, can enhance the predictive performance of the interpretable model by accurately assigning uncovered instances to the correct hypercube.

Brute prediction depends on the closest hypercube w.r.t. the query. The distance between a data point and a cube may be calculated according to more than one definition. In the following several strategies are defined, each one presenting a trade-off between computational complexity and expected predictive performance. In particular, we propose to assimilate the distance between a point and a cube to the Euclidean distance between the point and the *relevant points* of the cube. The selection of Euclidean distance derives from the need to maintain the highest possible degree of human interpretability for the SKE outputs. Indeed, the most natural method to assess distance for humans is in terms of “straight lines” between two points. Other commonly used metrics are the Manhattan distance or the Chebyshev distance. We briefly recall that the Manhattan distance between a pair of points is calculated as the sum of the absolute differences of the points’ Cartesian coordinates (“city block” strategy).

On the other hand, the Chebyshev distance between two points is the greatest distance amongst those calculated for each point dimension ("chessboard" strategy). In our opinion, these alternatives hinder the immediacy of the knowledge representation, even if they may be more computationally efficient.

In our proposed extension we consider the following points of the hypercube as relevant: the centre, barycentre, vertices, and edge points. This allows us to employ different approaches for brute prediction, such as centre-based, density-based, corner-based, and perimeter-based methods. Additionally, we introduce an additional majority criterion that disregards the identified hypercubes and instead relies solely on the average output observed for the instances provided during the extraction phase of the SKE technique. This criterion can be used as an alternative prediction strategy.

To establish the different strategies for brute prediction, we begin by formalising the concepts of a data set and a hypercube. We define a data set D as a collection of n-dimensional points, where n represents the number of input features in the data set. For the purpose of this definition, without loss of generality, we will ignore the output feature since the hypercubes created using SKE techniques are based on the dimensions of the input features alone. The domain of data set D, i.e. $Dom(D)$, is defined as the Cartesian product of the domains of each input feature f of D:

$$Dom(D) = Dom(f_1) \times Dom(f_2) \times \cdots \times Dom(f_n). \tag{1}$$

Hypercube-based SKE algorithms work upon continuous input features, therefore

$$Dom(f_i) \subseteq \mathbb{R} \qquad \forall i = 1, \ldots, n. \tag{2}$$

As a consequence

$$Dom(D) \subseteq \mathbb{R}^n. \tag{3}$$

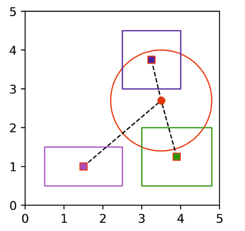
A hypercube H is defined as a portion of the input feature space:

$$H \subseteq D. \tag{4}$$

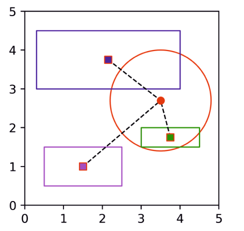
The corresponding domain is thus a subset of the domain of data set D. We denote with h_1, h_2, ..., h_n the n individual dimensions of the cube, with the following domains:

$$Dom(h_i) \subseteq Dom(f_i) \qquad \forall i = 1, \ldots, n. \tag{5}$$

It is worthwhile to notice that hypercubes (and their corresponding domains) are usually strict subsets of the data set (and its corresponding domain), except for the *surrounding cube*, defined as the cube enclosing all the instances of the data set. Therefore, this cube coincides with the data set and its domain coincides with the one of the data set.



(a) Expected behaviour.



(b) Issues with unbalanced cube sizes.

Fig. 1. Example of centre-based distance calculation.

3.1 Majority-Based Assignment

The simplest option to provide predictions for data instances belonging to uncovered input space regions is to consider a surrounding hypercube enclosing all the possible queries. This enables SKE techniques to exhibit a default behaviour when there are no cubes providing the needed prediction. To minimise the predictive error of this default prediction it is necessary to consider the most common output values observed in the whole data set. When performing classification tasks, it is possible to consider as default output the most common class label in the data set. Conversely, for regression tasks the output feature can be averaged over all the data points to provide a constant value. Alternatively, it is possible to express the output value as a linear function of the input features approximating the data point distribution within the whole data set.

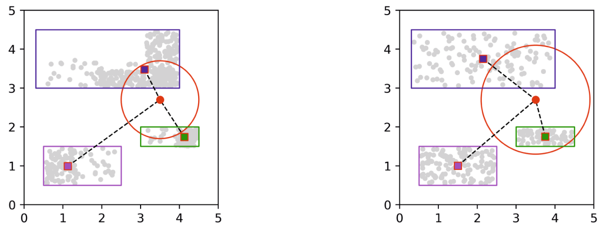
With this majority-based criterion brute predictions may be provided in constant time, regardless of the number of input features describing the data set, and without calculating any distance between queries and hypercubes found via SKE. However, the default output value is strongly subject to the data used to extract knowledge. For instance, if a balanced data set with 3 classes is randomly split into training and test sets and only the training set is used to extract knowledge (as usually done), the default value will be determined based on the class label distribution after the random train/test splitting, leading to arbitrary brute predictions.

3.2 Centre-Based Assignment

A slightly more complex solution, albeit with comparable computational complexity, resides in the calculation of the Euclidean distance between the query and the centre of each identified hypercube. The brute prediction is then provided based on the hypercube whose centre is the closest to the query.

We define the centre of a hypercube H as the multidimensional point whose coordinates are the centres of the cube's dimensions:

$$Centre(H) = (Centre(h_1), Centre(h_2), ..., Centre(h_n)). \tag{6}$$


(a) Expected behaviour.

(b) Issues with uniform density cubes.

Fig. 2. Example of density-based distance calculation.

We finally define the centre of a hypercube dimension h as the dimension midpoint:

$$Centre(h) = \frac{max(h) + min(h)}{2}. \tag{7}$$

The centre-based criterion requires the calculation and comparison of a single distance for each hypercube since the only relevant point is the cube centre. However, it may be not a proper strategy when there are cubes having very different sizes. In this case, small cubes are arbitrarily privileged since instances belonging to uncovered regions have more probability of being closer to the centre of small cubes than to those of large cubes.

An example of an expected centre-based assignment is reported in Fig. 1a, where the point belonging to the uncovered region (red point) is associated with the closest cube (the blue one). The issue due to high diversity in the cube sizes is reported in Fig. 1b, where the point is associated with the green hypercube, given the vicinity to its centre, even if the point is visibly closer to the blue hypercube.

3.3 Density-Based Assignment

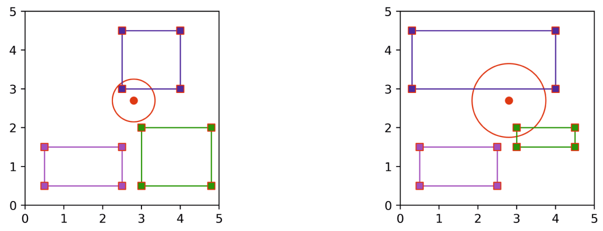
Centre-based brute predictions may be theoretically enhanced by adding awareness of the data set instance distribution within the identified hypercubes. In this case we move towards a density-based assignment of the uncovered queries since the brute prediction is based on the distance between a query and the barycentres of the cubes identified via SKE.

We define the barycentre of a hypercube H as the multidimensional point whose coordinates are the barycentres of the cube's dimensions:

$$Barycentre(H) = (Barycentre(h_1), Barycentre(h_2), ..., Barycentre(h_n)). \tag{8}$$

We finally define the barycentre of a hypercube dimension h as the mean value calculated for that dimension on the data set instances enclosed within the hypercube:

$$Barycentre(h) = \overline{h}. \tag{9}$$



(a) Expected behaviour. (b) Issues with unbalanced cube sizes.

Fig. 3. Example of corner-based distance calculation.

The density-based criterion for brute prediction has the same computational complexity as the centre-based criterion. In both cases, only a single distance calculation and comparison are required for each identified hypercube in the SKE technique. However, the density-based criterion may encounter drawbacks when it comes to assigning instances to hypercubes that have a uniform distribution of data points. In such cases, the barycentres of the hypercubes are equivalent to the centres, resulting in similar issues as those described for the centre-based brute prediction.

Examples of density-based assignments are shown in Fig. 2a (expected assignment) and Fig. 2b (incorrect assignment due to hypercubes with uniform density).

3.4 Corner-Based Assignment

Given that manual assignments of uncovered instances to hypercubes performed by human users would take into account the distance of the data point to the edges of the cube, we formalise accordingly a corner-based criterion considering the cube vertices.

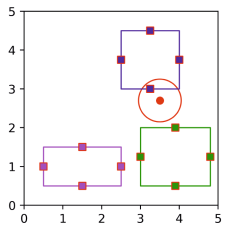
We define the corners of a hypercube H as the set of points obtained via the Cartesian product of the sets of corners corresponding to the individual cube's dimension:

$$Corners(H) = \bigtimes_{i=1}^{n} Corners(h_i). \tag{10}$$

We finally define the corners of a hypercube dimension h as the set containing its minimum and maximum values:

$$Corners(h) = \{min(h), max(h)\}. \tag{11}$$

The computational complexity of corner-based brute prediction is no longer constant. For each assignment, it is necessary to calculate the distance between the uncovered instance and each corner of each identified hypercube. Therefore,


(a) Expected behaviour.

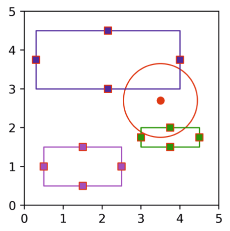

(b) Issues with unbalanced cube sizes.

Fig. 4. Example of midpoint-based distance calculation.

the complexity of corner-based brute prediction is directly related to the dimensionality of the hypercubes, which corresponds to the number of input features in the data set, denoted as n. The number of corners in an n-dimensional hypercube is equal to 2^n. This means that as the dimensionality of the hypercube increases, the number of corners and the computational complexity of corner-based brute prediction also increase. However, corner-based brute prediction remains feasible even with high-dimensional data sets. It is worth noting that when using the corner-based strategy, there can be challenges when assigning instances to hypercubes of different sizes. In such cases, smaller cubes may be privileged, leading to potential issues in the assignment process. The assignment may become arbitrary and dependent on the positioning of the hypercubes within the input space. Examples of corner-based assignments are reported in Figures 3a and 3b, which demonstrate the expected behavior and highlight the issues that can arise when hypercubes of significantly different sizes are present.

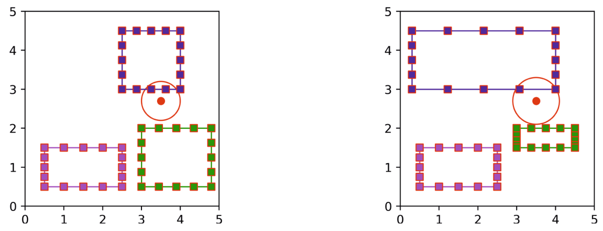
3.5 Midpoint-Based Assignment

Alternatively, other relevant multidimensional points laying on the hypercube edges may be selected to compute the Euclidean distances. For instance, one can use edge midpoints instead of vertices.

We define the midpoints of a hypercube H as the set obtained through the union of the Cartesian products of an edge midpoint with the corners of all the other cube edges:

$$Midpoints(H) = \bigcup_{i=1}^{n} \left\{ \{Centre(h_i)\} \times \bigtimes_{j \in \{1,\ldots,n\} \setminus \{i\}} Corners(h_j) \right\}. \quad (12)$$

The computational complexity of midpoint-based brute prediction is slightly higher than that of corner-based assignments. This strategy involves calculating the distance between the uncovered instance and every midpoint of each identified hypercube. The number of midpoints in an n-dimensional hypercube is equal to the number of edges, which is given by $n \cdot 2^{n-1}$. However, it is worth noting that the negative effects on computational complexity become noticeable



(a) Expected behaviour.

(b) Expected behaviour with unbalanced cube sizes.

Fig. 5. Example of perimeter-based distance calculation, $p = 5$.

when the dimensionality of the hypercube exceeds 6. Therefore, midpoint-based assignments can still be computationally feasible for data sets with dimensions up to a certain threshold. Similar to corner-based assignments, midpoint-based assignments also face challenges when dealing with hypercubes of significantly different sizes. The issues arising from these cubes are shared between the two strategies, as shown in Fig. 4.

3.6 Perimeter-Based Assignment

The notion of midpoint-based assignment may be relaxed in order to enable a more fine-grained sampling of edge points and to overcome the issues of cubes having different sizes. We then introduce the p parameter to indicate how many equispaced points have to be selected on edges and modify Eq. (12) accordingly:

$$Perimeter(H, p) = \bigcup_{i=1}^{n} \left\{ Equispaced(h_i, p) \times \bigtimes_{j \in \{1,\ldots,n\} \setminus \{i\}} Corners(h_j) \right\}, \tag{13}$$

where $Equispaced(h, p)$ denotes the set of p equispaced points for the hypercube dimension h.

The accuracy of perimeter-based brute prediction is greater with greater values of p, however, the computational complexity grows together with p. Indeed, the number of relevant points identified for a single n-dimensional cube is equal to $p \cdot n \cdot 2^{n-1}$ (i.e., p points per edge). This value is actually reduced in practice, given that duplicate points corresponding to the cube edges are ignored. An n-dimensional cube has $(n-1) \cdot 2^n$ duplicate points. The number of relevant points without duplicates is thus $p \cdot n \cdot 2^{n-1} - (n-1) \cdot 2^n$, equivalent to $(p-2) \cdot n \cdot 2^{n-1} + 2^n$. Perimeter-based assignments are exemplified in Fig. 5.

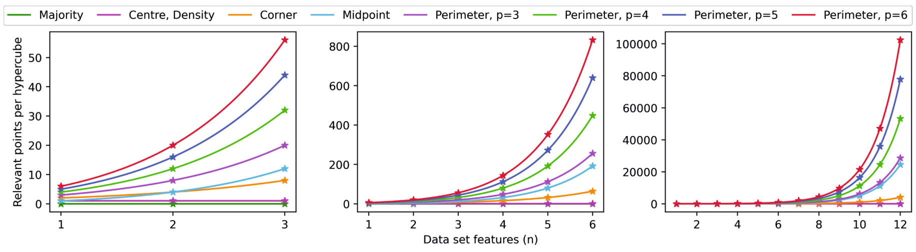


Fig. 6. Computational complexity of the proposed vicinity-based assignments.

It is worthwhile to notice that with $p = 1$ this strategy corresponds to the midpoint-based brute prediction, whereas with $p = 2$ it corresponds to the corner-based one.

3.7 On the Computational Complexity

The computational complexity of each brute prediction strategy described in this work is reported in Fig. 6. The figure shows the complexity of perimeter-based assignments for different values of the p parameter, up to 6.

Unfortunately, there is an important trade-off between the quality of the brute prediction and the corresponding computational complexity. Indeed, as described in this section, the simplest and fastest strategies to be applied are the most prone to provide predictive errors, e.g., when the sizes of hypercubes are strongly unbalanced. On the other hand, perimeter-based predictions have more probability to give the correct results. However, this strategy becomes quite unfeasible for $p = 3$ when $n > 6$, given that the overall amount of relevant points *of a single cube* would be greater than 576 (corresponding to the calculation and comparison of as many distances for each query to predict). For the same reason the perimeter-based strategy is not recommended with $p \in \{4, 5\}$ if $n > 5$ and with $p = 6$ if $n > 4$.

Obviously, the complexity of the adopted strategy also depends on the number of cubes identified via the SKE technique, given that the reported measurements have to be multiplied by the number of cubes. SKE algorithms usually output a limited amount of hypercubes to preserve human readability and therefore the overall complexity of the vicinity-based assignments is not strongly altered. However, there may be applications where the hypercube amount is bound to the domain itself and it has a heavy impact on the computational complexity of vicinity-based brute prediction, e.g., classification tasks for handwritten digits or characters. Even by assuming an optimal hypercubic partitioning of the input feature space, resulting in only a single cube per possible output class, 10 or 26 different cubes are identified, respectively, thus causing a non-negligible impact on the overall computational complexity.

4 Experiments

To evaluate the effectiveness of our proposed extension, we performed a number of experiments with the aid of the PSyKE[2] Python framework [3,21,23,26].

We selected the Iris data set[3] [6] to exemplify our proposed vicinity-based brute prediction strategies since it can be easily visualised as a bidimensional projection. In particular, we privileged the petal length and width input features, given that these are the most relevant input features to perform the classification.

The experimental setup is the following. The sepal length and width input features have been first removed from the data set. Then, the 150 available data instances were split into 2 halves, one to train an opaque model and extract knowledge, the other to test the predictive performance of both the model and the knowledge.

A k-nearest neighbours with $k = 7$ (7-NN) has been selected as opaque underlying model for the GridEx hypercube-based knowledge extractor. Data samples and decision boundaries of the 7-NN and the GridEx extractor are shown in Fig. 7.

From Fig. 7c it is possible to notice that GridEx identifies 3 different hypercubes, one per output class of the data set. The cubes do not cover the whole input feature space, yet they enclose the majority of data samples. Only 2 instances of Virginica Iris are outside the boundaries of the cubes (green stars with blue and magenta contour in Fig. 7c) and thus they cannot be predicted through the interpretable model provided by GridEx.

In Fig. 7 the proposed vicinity-based brute predictions are exemplified. In particular, Fig. 7d shows the majority-based assignment of the 2 uncovered instances. Since the random train/test splitting produced a training set where the Versicolor output class label is predominant, both uncovered instances are wrongly classified as Versicolor Iris. This very simple strategy is thus easy to compute but not accurate from a predictive standpoint.

Figure 7e and 7f reports the assignment obtained according to the centre- and density-based strategies, respectively. In both cases only one uncovered instance is correctly classified as Virginica Iris (the magenta one), whereas the other is misclassified as Versicolor Iris (the blue one).

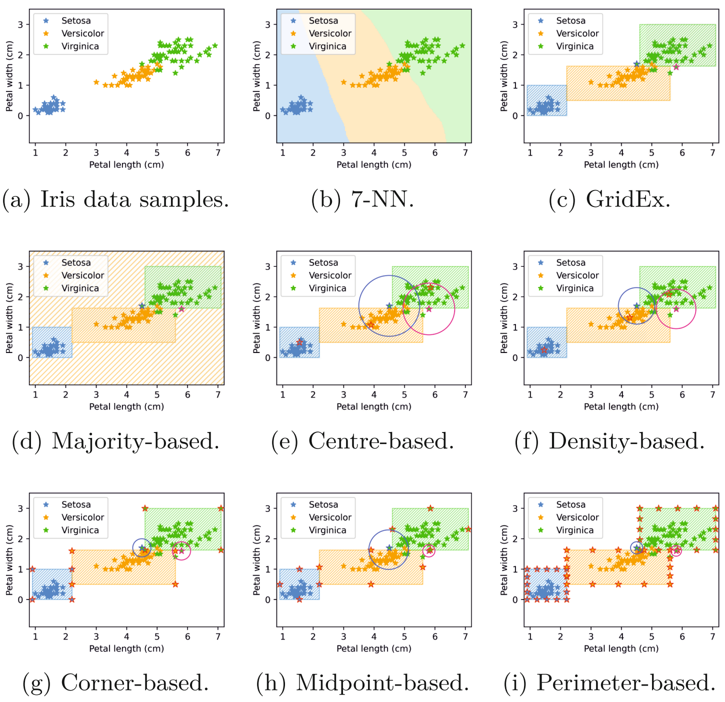
The inverse situation is present by adopting a corner-based brute prediction, as depicted in Fig. 7g. According to the distance between uncovered instances and cubes' corners, the blue instance is incorrectly associated with the Versicolor class and the magenta one is correctly classified as a Virginica Iris.

Both instances are misclassified by adopting midpoint-based assignments (cf. Fig. 7h).

Finally, Fig. 7i shows the effectiveness of perimeter-based brute prediction. Indeed, both uncovered instances are correctly classified by adopting this strategy with $p = 5$.

[2] https://github.com/psykei/psyke-python.

[3] https://archive.ics.uci.edu/dataset/53/iris.



(a) Iris data samples. (b) 7-NN. (c) GridEx.

(d) Majority-based. (e) Centre-based. (f) Density-based.

(g) Corner-based. (h) Midpoint-based. (i) Perimeter-based.

Fig. 7. Example of vicinity-based assignments for GridEx on the Iris data set.

4.1 Experiments on Real World Data Sets

A quantitative assessment of the predictive performance obtained for the distance-based brute prediction on the Iris data set is reported in Table 1. In this case the data set has been used without removing input features. The same training/test splitting strategy (50% + 50%) has been adopted. A new 7-NN has been trained accordingly, achieving an accuracy score of 0.96. A set of non-exhaustive hypercube-based SKE techniques (i.e., Iter and 2 instances of GridEx, having different parameters) are compared with 2 instances of CReEPy, used as benchmark for hypercube-based SKE methods complete by design. The parameters of each algorithm are reported in the table. In particular, Iter requires the size of the single updates, the maximum number of iterations (it), the number of starting points, and an error threshold (θ). GridEx requires a splitting strategy, a maximum depth (δ) and an error threshold (θ). We selected for GridEx adaptive strategies based on the input feature relevance: we let the 2 algorithm instances perform 5 or 8 slices along features with relevance greater than 0.5 or 0.85, respectively, and only a single slice along all the others. Finally, CReEPy requires an underlying explainable clustering technique, a maximum depth (δ) and an error threshold (θ).

Table 1. Predictive performance of brute predictions for the Iris data set.

	k-NN	Iter	GridEx	GridEx	CReEPy	CReEPy
Parameters	$k = 7$	update = 0.1 $it = 100$ points = 1 $\theta = 0.1$	adaptive split if relev. < 0.5: 1 split else 5 $\delta = 1$ $\theta = 0.1$	adaptive split if relev. < 0.85: 1 split else 8 $\delta = 1$ $\theta = 0.1$	Clustering: ExACT $\delta = 2$ $\theta = 0.1$	Clustering: CREAM $\delta = 2$ $\theta = 0.1$
Accuracy	0.96	0.97	0.96	0.90	0.92	0.93
Rules		3	6	3	3	3
Completeness		0.93	0.92	0.92	1.00	1.00
Majority		0.91	0.88	0.83		
Centre		0.97	0.96	0.91		
Density		0.97	0.96	0.91		
Corner		0.97	0.96	0.91		
Midpoint		0.97	0.96	0.91		
Perimeter, $p = 3$		0.97	0.96	0.91		
Perimeter, $p = 5$		0.97	0.96	0.91		

For each SKE instance the fidelity w.r.t. the 7-NN expressed as classification accuracy, the number of extracted rules and the completeness expressed as percentage of test set covered by the identified hypercubes are reported in Table 1. For the non-exhaustive SKE techniques, several vicinity-based brute prediction strategies have been tested and the corresponding results are reported in the same table.

From the table, it is possible to notice that CReEPy outperforms the other techniques on 2 different dimensions, indeed it has the smallest amount of rules and the highest completeness. However, thanks to the proposed brute prediction strategies, Iter and GridEx may achieve complete coverage of the input feature space as well. As a result, Iter becomes the best algorithm in the examined pool, since it maximises both the completeness and the accuracy of its predictions with only 3 extracted rules. As for GridEx, an instance is able to outperform CReEPy in terms of accuracy but produces a double amount of rules, whereas the other exhibits the same rule amount but smaller classification accuracy. The proposed extension to draw predictions for uncovered instances is thus effective in obtaining better overall results.

We conclude this experimental section with a regression case study on the Combined Cycle Power Plant (CCPP) data set [28]. A random forest (RF) regressor based on 20 decision trees (DT) with unbounded depth (δ) and leaf amount (λ) has been selected as opaque model. The pool of SKE techniques is composed of GridEx, GridREx, CReEPy adopting CREAM clustering, and 2 different instances of Iter. We omitted the results of CReEPy adopting ExACT clustering, since it showed the same input space partitioning obtained by adopting CREAM, with the same performance measurements, by setting $\delta = 3$. Results are reported in Table 2, where predictive performance is expressed through the R^2 score.

Table 2. Predictive performance of brute predictions for the CCPP data set.

	RF	ITER	ITER	GridEx	GridREx	CREEPY
Parameters	DT = 20 $\lambda = \infty$ $\delta = \infty$	upd. = 0.03 $it = 100$ points = 1 $\theta = 10$	upd. = 0.07 $it = 150$ points = 2 $\theta = 10$	adaptive split if relev. < 0.7: 1 split else 3 $\delta = 1$ $\theta = 1$	adaptive split if relev. < 0.7: 1 split else 3 $\delta = 1$ $\theta = 1$	CREAM clust $\delta = 2$ $\theta = 0.1$ output: linear functions
R^2	0.96	0.62	0.86	0.98	0.93	0.97
Rules		4	13	3	9	4
Completeness		0.58	0.93	0.99	0.99	1.00
Majority		0.31	0.77	0.97	0.93	
Centre		0.65	0.86	0.98	0.93	
Density		0.61	0.84	0.98	0.93	
Corner		0.60	0.81	0.98	0.93	
Midpoint		0.62	0.87	0.98	0.93	
Perimeter, $p = 3$		0.62	0.87	0.98	0.93	
Perimeter, $p = 5$		0.63	0.86	0.98	0.93	

In this case study GridEx is superior to CREEPY from the predictive performance and rule amount standpoints. However, only CREEPY provides complete knowledge. Our proposed extension enables GridEx to achieve 100% completeness as well. GridEx empowered with brute prediction capabilities outperforms CREEPY and it is thus the best SKE technique in the pool.

Amongst all the proposed vicinity-based strategies, from our experiments, the majority-based criterion appears to be the simplest but also the least performing in terms of prediction accuracy. Conversely, the other alternatives do not exhibit very noticeable differences in the measured predictive performance. For this reason we suggest applying one of the least computationally expensive strategies to empower non-exhaustive hypercube-based SKE techniques.

5 Conclusions

In this paper, we introduce a vicinity-based extension for SKE techniques that ensures 100% completeness in predictions. This extension can be applied to any hypercube-based SKE method and offers flexibility to users in selecting the desired trade-off between computational complexity and predictive performance. We define different strategies based on *vicinity* for the extension, allowing users to tailor the approach to their specific needs. Additionally, we provide an analytical study of the computational complexity associated with the proposed extension. Furthermore, we present experimental results that demonstrate the effectiveness of the extension in practical applications.

Our future work aims to enhance the selection of relevant points within hypercubes to achieve smaller sets of points while maintaining comparable or even improved predictive performance with reduced computational complexity.

Specifically, we plan to consider points on the surface of the hypercubes and to develop mechanisms for coarse-grained sampling of relevant points in low-importance regions of the cubes, such as cube edges near the boundaries of the data set domain or perimeter sampling for very small cubes. These advancements will further refine and optimise the extension of SKE techniques.

Acknowledgments. This work has been supported by European Union's Horizon Europe AEQUITAS research and innovation programme under grant number 101070363.

References

1. Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowl. Based Syst. **8**(6), 373–389 (1995). https://doi.org/10.1016/0950-7051(96)81920-4
2. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. CRC Press, Boca Raton (1984)
3. Calegari, R., Sabbatini, F.: The PSyKE technology for trustworthy artificial intelligence. In: Dovier, A., Montanari, A., Orlandini, A. (eds.) XXI International Conference of the Italian Association for Artificial Intelligence, AIxIA 2022, Udine, Italy, November 28 – December 2, 2022, Proceedings, vol. 13796, pp. 3–16 (2023). https://doi.org/10.1007/978-3-031-27181-6_1
4. Craven, M.W., Shavlik, J.W.: Extracting tree-structured representations of trained networks. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems 8. Proceedings of the 1995 Conference, pp. 24–30. The MIT Press (1996). https://papers.nips.cc/paper/1152-extracting-tree-structured-representations-of-trained-networks.pdf
5. Eineborg, M., Boström, H.: Classifying uncovered examples by rule stretching. In: Rouveirol, C., Sebag, M. (eds.) Inductive Logic Programming, 11th International Conference, ILP 2001, Strasbourg, France, September 9–11 2001, Proceedings. LNCS, vol. 2157, pp. 41–50. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44797-0_4
6. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugenics **7**(2), 179–188 (1936). https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x, https://doi.org/10.1111/j.1469-1809.1936.tb02137.x
7. Garcez, A.S.d., Broda, K., Gabbay, D.M.: Symbolic knowledge extraction from trained neural networks: a sound approach. Artif. Intell. **125**(1–2), 155–207 (2001)
8. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 1–42 (2018). https://doi.org/10.1145/3236009
9. Huysmans, J., Baesens, B., Vanthienen, J.: ITER: an algorithm for predictive regression rule extraction. In: Tjoa, A.M., Trujillo, J. (eds.) DaWaK 2006. LNCS, vol. 4081, pp. 270–279. Springer, Heidelberg (2006). https://doi.org/10.1007/11823728_26
10. Kenny, E.M., Ford, C., Quinn, M., Keane, M.T.: Explaining black-box classifiers using post-hoc explanations-by-example: the effect of explanations and error-rates in XAI user studies. Artif. Intell. **294**, 103459 (2021). https://doi.org/10.1016/j.artint.2021.103459

11. Konig, R., Johansson, U., Niklasson, L.: G-REX: a versatile framework for evolutionary data mining. In: 2008 IEEE International Conference on Data Mining Workshops (ICDM 2008 Workshops), pp. 971–974 (2008). https://doi.org/10.1109/ICDMW.2008.117
12. Rocha, A., Papa, J.P., Meira, L.A.A.: How far do we get using machine learning black-boxes? Int. J. Pattern Recogn. Artif. Intell. **26**(02), 1261001-(1–23) (2012). https://doi.org/10.1142/S0218001412610010
13. 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). https://doi.org/10.1038/s42256-019-0048-x
14. Sabbatini, F., Calegari, R.: Symbolic knowledge extraction from opaque machine learning predictors: GridREx & PEDRO. In: Kern-Isberner, G., Lakemeyer, G., Meyer, T. (eds.) Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31–August 5, 2022 (2022). https://doi.org/10.24963/kr.2022/57. https://proceedings.kr.org/2022/57/
15. Sabbatini, F., Calegari, R.: Bottom-up and top-down workflows for hypercube- and clustering-based knowledge extractors. In: Proceedings of the V International Workshop on Explainable and Transparent AI and Multi-Agent Systems, EXTRAAMAS 2023, London, UK, 29 May 2023, vol. 14127. Springer, Cham. (2023, to appear). https://doi.org/10.1007/978-3-031-40878-6_7
16. Sabbatini, F., Calegari, R.: Explainable clustering via ExACT. In: Proceedings of the II International Workshop on Knowledge Diversity, KoDis 2023, Rhodes, Greece, 2–8 September 2023 (2023). https://ceur-ws.org/Vol-3548/paper3.pdf
17. Sabbatini, F., Calegari, R.: Explainable clustering with CREAM. In: Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, pp. 593–603 (2023). https://doi.org/10.24963/kr.2023/58
18. Sabbatini, F., Calegari, R.: The ICE score to evaluate symbolic knowledge quality. In: Proceedings of the XXXVIII Annual AAAI Conference on Artificial Intelligence, AAAI24, Vancouver, Canada, 20–27 February 2024 (2023, submitted to)
19. Sabbatini, F., Calegari, R.: On the evaluation of the symbolic knowledge extracted from black boxes. In: AAAI 2023 Spring Symposium Series, San Francisco, California (2023, to appear)
20. Sabbatini, F., Calegari, R.: Unveiling opaque predictors via explainable clustering: the CReEPy algorithm. In: Proceedings of the 2nd Workshop on Bias, Ethical Al, Explainability and the role of Logic and Logic Programming, BEWARE-23, Rome, Italy, November 6, 2023, (2023, to appear)
21. Sabbatini, F., Ciatto, G., Calegari, R., Omicini, A.: On the design of PSyKE: a platform for symbolic knowledge extraction. In: Calegari, R., Ciatto, G., Denti, E., Omicini, A., Sartor, G. (eds.) WOA 2021–22nd Workshop "From Objects to Agents". CEUR Workshop Proceedings, vol. 2963, pp. 29–48. Sun SITE Central Europe, RWTH Aachen University (2021). https://ceur-ws.org/Vol-2963/paper14.pdf, 22nd Workshop "From Objects to Agents" (WOA 2021), Bologna, Italy, 1–3 September 2021. Proceedings (2021)
22. Sabbatini, F., Ciatto, G., Calegari, R., Omicini, A.: Hypercube-based methods for symbolic knowledge extraction: towards a unified model. In: Ferrando, A., Mascardi, V. (eds.) WOA 2022–23rd Workshop "From Objects to Agents", CEUR Workshop Proceedings, Sun SITE Central Europe, RWTH Aachen University, vol. 3261, pp. 48–60 (2022). https://ceur-ws.org/Vol-3261/paper4.pdf

23. Sabbatini, F., Ciatto, G., Calegari, R., Omicini, A.: Symbolic knowledge extraction from opaque ML predictors in PSyKE: platform design experiments. Intelligenza Artificiale **16**(1), 27–48 (2022). https://doi.org/10.3233/IA-210120
24. Sabbatini, F., Ciatto, G., Calegari, R., Omicini, A.: Towards a unified model for symbolic knowledge extraction with hypercube-based methods. Intelligenza Artificiale **17**(1), 63–75 (2023). https://doi.org/10.3233/IA-230001
25. Sabbatini, F., Ciatto, G., Omicini, A.: GridEx: an algorithm for knowledge extraction from black-box regressors. In: Calvaresi, D., Najjar, A., Winikoff, M., Främling, K. (eds.) Explainable and Transparent AI and Multi-Agent Systems. Third International Workshop, EXTRAAMAS 2021, Virtual Event, 3–7 May 2021, Revised Selected Papers, LNCS, vol. 12688, pp. 18–38. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-82017-6_2
26. Sabbatini, F., Ciatto, G., Omicini, A.: Semantic Web-based interoperability for intelligent agents with PSyKE. In: Calvaresi, D., Najjar, A., Winikoff, M., Främling, K. (eds.) Explainable and Transparent AI and Multi-Agent Systems, LNCS, vol. 13283, pp. 124–142. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-15565-9_8
27. Salzberg, S.: A nearest hyperrectangle learning method. Mach. Learn. **6**, 251–276 (1991). https://doi.org/10.1023/A:1022661727670
28. Tüfekci, P.: Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods. Int. J. Electr. Power Energy Syst. **60**, 126–140 (2014). https://www.sciencedirect.com/science/article/pii/S0142061514000908, https://doi.org/10.1016/j.ijepes.2014.02.027