

# Unveiling Opaque Predictors via Explainable Clustering: The CReEPy Algorithm<sup>\*</sup>

Federico Sabbatini<sup>1,\*</sup>, Roberta Calegari<sup>2</sup>

<sup>1</sup>Department of Pure and Applied Sciences, University of Urbino Carlo Bo

<sup>2</sup>Department of Computer Science and Engineering (DISI), Alma Mater Studiorum–University of Bologna

## Abstract

Machine learning black boxes, as deep neural networks, are often hard to explain because their predictions depend on complicated relationships involving a huge amount of internal parameters and input features. This opaqueness from the human perspective makes their predictions not trustable, especially in critical applications. In this paper we tackle this issue by introducing the design and implementation of CReEPy, an algorithm performing symbolic knowledge extraction based on explainable clustering. In particular, CReEPy relies on the underlying clustering performed by the ExACT or CREAM procedures to provide human-interpretable Prolog rules mimicking the behaviour of the opaque model. Experiments to assess both the human readability and the predictive performance of the proposed algorithm are discussed here, using existing state-of-the-art techniques as benchmarks for the comparison.

## Keywords

Explainable clustering, Explainable artificial intelligence, Symbolic knowledge extraction, PSyKE

## 1. Introduction

In recent years, there has been a growing demand for transparency, particularly in critical domains [1, 2]. This demand has led to a lack of trust among humans in predictions obtained from machine learning (ML) models that lack interpretability. Such models are often referred to as *opaque* or *black boxes* (BBs) due to their inscrutability. While complex ML models tend to offer superior predictive performance, they pose challenges when it comes to human inspection. Consequently, the use of opaque models for high-stakes decisions necessitates the derivation of human-intelligible knowledge to ensure accountability and understanding.

To not renounce the impressive predictive capabilities of ML models, many strategies to obtain explainable behaviours have been proposed in the literature [3, 4], for instance, the adoption of interpretable ML predictors [5] or mechanisms to reverse-engineer the predictors' behaviour [6]. Symbolic knowledge-extraction (SKE) techniques are exploited to this end, acting in a post-processing phase to extract interpretable knowledge out of a BB predictor.

---

*2nd Workshop on Bias, Ethical AI, Explainability and the role of Logic and Logic Programming, BEWARE-23, co-located with AlxIA 2023, Roma Tre University, Roma, Italy, November 6, 2023*


<sup>\*</sup> Original research paper.

<sup>\*</sup> Corresponding author.

✉ f.sabbatini1@campus.uniurb.it (F. Sabbatini); roberta.calegari@unibo.it (R. Calegari)

🆔 0000-0002-0532-6777 (F. Sabbatini); 0000-0003-3794-2942 (R. Calegari)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Following the current development on the SKE field [7], we developed CReEPy, a new general-purpose knowledge-extraction procedure based on interpretable clustering and applicable to any kind of BB predictor. CReEPy is built upon ExACT and CREAM and it pedagogically explains BBs performing classification or regression tasks and operating on continuous input features. CReEPy proves the effectiveness of exploiting explainable clustering to achieve the interpretability of BBs. Indeed, it enables the extraction of more concise and accurate explanations compared to analogous state-of-the-art techniques.

The paper is organised as follows: Section 2 introduces background information on the topics discussed here and related works present in the literature. Section 3 describes the CReEPy algorithm. Experiments and benchmark comparisons are discussed in Section 4. Finally, conclusions are drawn in Section 5.

## 2. Related Works

### 2.1. Symbolic Knowledge Extraction

SKE consists of obtaining human-interpretable rules out of BB predictors by means of a surrogate, explainable model that is able to mimic the BB, named in this context *underlying model*. The underlying model may be a classifier, a regressor, a clustering technique, or any other opaque predictor. The mimicking capabilities of the surrogate model are assessed via the comparison of the outputs provided by the underlying and surrogate models w.r.t. the same inputs. SKE techniques are currently applied in a wide range of contexts [8, 9, 10, 11, 12, 13, 14].

The construction of the surrogate predictor may be performed in a decompositional or pedagogical way [15]. In the former case, the BB kind and internal structure are considered, so these algorithms are not general and can be applied only to a subset of BBs, e.g., RefAnn [16] accepts as underlying predictors only neural networks having a single hidden layer. On the other hand, pedagogical techniques only consider the underlying BB input/output relationship and thus they are more general and present no constraints on the BB type and complexity.

In the following we provide a brief description of the SKE algorithms chosen as benchmarks for the experiments presented in this work.

#### 2.1.1. ITER

ITER [17] is a pedagogical knowledge-extraction algorithm explicitly designed for black-box regressors. It extracts knowledge in the form of rule lists while imposing no constraint on the nature, structure, or training of the underlying opaque model.

To extract rules, the ITER algorithm steps through the creation and iterative expansion of several disjoint hypercubes, covering the whole input space the regressor has been trained upon. In other words, ITER accepts as input a regressor and the data set used for its training, then iteratively partitions the input feature space following a bottom-up strategy.

At the end of the process, each partition is converted into a human-interpretable rule associated with a constant output value.

### 2.1.2. GridEx and GridREx

The GridEx algorithm [18] is a pedagogical technique performing symbolic knowledge extraction from BBs designed for regression tasks. Thanks to the generalisation proposed in [19, 20] it can be also applied to explain classifiers. In both cases, data sets have to be described by continuous features. It is inspired to ITER, with the aim of removing the issues deriving from its possible slow convergence as well as the small input space coverage and fidelity when dealing with high-dimensional data sets. GridEx satisfies this goal by relying on a top-down partitioning strategy, thus achieving good results in terms of both the number of extracted rules and corresponding predictive performance w.r.t. the underlying BB and the data.

The partitioning strategy adopted by GridEx consists of the recursive input feature space splitting into smaller subregions according to a similarity threshold. At the end of the partitioning, each region is translated into a human-readable rule, having preconditions describing the region and a postcondition representing the associated output value, which is a constant obtained by averaging the underlying model predictions for the samples included in the region.

Unfortunately, in some real-world applications, the undesired discretisation introduced by the constant outputs of GridEx may hinder the predictive performance of the extractor. GridREx [21] overcomes this issue by training a linear model inside each identified hypercubic region. Linear models are fitted on the instances contained in the corresponding hypercubes and each cube is associated with a rule having a set of conditions on the input variables as antecedent part, equally to ITER and GridEx, and a linear combination of the input variables (given by the linear model) as consequent part. As a result, output predictions given by the extracted rules are no more averaged output values of the samples contained in the corresponding hypercubes, but more accurate linear equations.

A disadvantage shared by GridEx and GridREx is that they perform a *symmetric* partitioning – i.e., during a given iteration, they split each input dimension in a given number of congruent partitions. This strategy may lead to suboptimal solutions when applied to real-world data sets.

### 2.1.3. CART

CART [22] is not properly a SKE technique, since it is based on the induction of binary decision trees on data set instances. However, it may be applied as well to the output of a BB predictor to obtain a decision tree representing the BB behaviour. Starting from the tree, it is straightforward to extract human-comprehensible rules by converting each possible path from the tree root to the leaves into a logic rule. CART can be applied to both BB classifiers and regressors, however also in this case the output value is a constant, so predictions suffer from an undesired discretisation when used in regression tasks.

## 2.2. Explainable Clustering via ExACT and CREAM

ExACT [23] is an algorithm performing explainable clustering. It combines together the aggregation strategies proper of traditional clustering techniques and the cluster assignment via decision trees as done by other explainable or interpretable clustering procedures. For this reason with ExACT it is possible to obtain explainable clusters by inducing a top-down decision tree over the training data according to a strictly hierarchical strategy. Indeed, identified clusters

---

**Algorithm 1** CReEPy pseudocode

---

**Require:** underlying clustering parameters  $\Pi$

**Require:** input feature importance set  $\Phi$

**Require:** feature importance threshold  $\Theta$

```
1: function CReEPy( $P, D$ )
2:    $D' \leftarrow$  CREATEDATASET( $P, D$ )
3:    $regions \leftarrow$  CLUSTERING( $\Pi, D'$ )
4:   return  $\bigcup_{r \in regions} \{ REGIONTORULE(r) \}$ 

5: function REGIONTORULE( $region$ )
6:   return a Prolog rule describing  $region$  in terms of its relevant features, by comparing  $\Phi$  and  $\Theta$ 

7: function CREATEDATASET( $P, D$ )
8:   return data set  $D$  with output feature predicted by  $P$ 
```

---

have the peculiarity of being concentric. The strategy adopted for the tree’s internal nodes is to use hypercubic splits to separate whole clusters of data while avoiding the presence of instances from multiple clusters inside the same hypercubic region.

Explainability is obtained by approximating each identified cluster with a hypercube. The concentric nature of the ExACT’s hierarchical approximations enables the creation of a global interpretable clustering in the form of a rule list, where each cluster is simply expressed through a rule having a single hypercube inclusion constraint, starting from the innermost cluster through the outermost. The same structure may be used to provide local explanations for single clustering assignments.

CREAM [24] extends ExACT by providing a more complex splitting strategy based on the iterative greedy minimisation of the predictive error measured for each possible split.

### 3. Symbolic Knowledge Extraction via Explainable Clustering

In this section we propose the design and implementation of a new knowledge-extraction technique, named CReEPy, that is able to obtain human-interpretable rules in Prolog syntax out of BB models of any sort and applicable to both classification or regression tasks. Following the idea proposed in [25], CReEPy performs the knowledge extraction by applying a preliminary interpretable clustering technique (i.e., ExACT or CREAM) to the training data, where the output feature is substituted with the opaque predictions. In the experiments reported here the ExACT procedure has been employed.

#### 3.1. The CReEPy Algorithm

CReEPy (Clustering-based REcursive Extraction as a PYramid) is a general-purpose pedagogical SKE technique applicable to any kind of BB predictor performing classification or regression tasks. It relies on the cluster approximation performed by ExACT or CREAM with the goal of providing human-readability to the underlying BB predictions and it is resumed in Algorithm 1.

CReEPy has been envisaged to be unbounded w.r.t. the underlying clustering. For this reason, it may be executed together with different clustering techniques if these provide hypercubic

input space approximations. Furthermore, CReEPy may be extended in the future to become compliant with other tree-based clustering approaches, since they basically slice the input feature space with cuts that are perpendicular to the axes and each path from the tree root to a leaf may be translated into a hypercube.

Being explicitly designed to work in synergy with ExACT and CREAM, CReEPy produces logic knowledge in the form of a theory of Prolog clauses (examples are shown in the experiment section). The theory mimics the decisions of the underlying BB model and each clause is related to an approximated cluster identified with ExACT or CREAM, enhancing its readability. Since these clusters are hierarchical cubes and difference cubes, thus defined as interval inclusions and exclusions, Prolog theories are particularly suited, due to the fact that clauses are *ordered*. As a consequence, it is possible to associate each clause only to preconditions referring to the *inclusion* in a hypercube, assuming as true the *exclusion* from all the cubes described by the preceding clauses. The expressiveness of this semantics is thus exploited at its peak by ordering the Prolog rules starting from the one associated with the innermost hypercubic region and then following the hierarchy up to the outermost region—equivalent to the surrounding cube of the data set at hand.

### 3.2. User-Defined Parameters

CReEPy's theory readability depends on the extracted rule amount and may be adjusted by tuning the underlying clustering instance parameters. Readability also depends on the number of preconditions per clause. ExACT and CREAM assign a precondition to each input dimension, i.e., an interval inclusion constraint for each input feature. Therefore, in the default version of CReEPy each Prolog clause has  $n$  preconditions for  $n$ -dimensional data sets. This may be limiting when dealing with high-dimensional data sets. For such a reason, users can provide CReEPy with the *input feature relevance* and a corresponding threshold, to limit the rule preconditions to the only features with relevance greater than the threshold. We highlight here that the input feature relevance is calculated outside CReEPy, so users are not bounded to a specific method, as far as they provide the feature relevance set normalised in the  $[0, 1]$  interval. A relevance score for each input feature is mandatorily required. A suitable and fast solution to obtain these scores can be found within the Python Scikit-Learn library.<sup>1</sup> It is worthwhile to point out that the feature relevance threshold does not affect the underlying clustering, but only the translation into Prolog rules performed by CReEPy starting from the tree provided by ExACT or CREAM (cf. REGIONTORULE procedure in Algorithm 1).

The translation into Prolog rules is executed according to the following criteria:

- for each leaf of the tree identified via the underlying clustering technique a rule is created;
- individual rules are *if-then* logic rules where the conditional part is a conjunction of interval inclusion constraints on the input features and the corresponding action is a constant value (e.g., a class label or a number) or a linear combination of the input variables;
- constraints are defined in the internal nodes of the tree;

---

<sup>1</sup>cf. [https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature\\_selection](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_selection)

- actions are described in the leaves of the tree;
- all variables having relevance smaller than the user-defined threshold are removed from the conditional part of the logic rules;
- the resulting rules are converted into a theory having Prolog format, both human- and agent-interpretable.

From a predictive perspective, the quality of CReEPy’s rules can be assessed via standard scores generally adopted for ML classification and regression tasks, e.g., accuracy and  $F_1$  score for the former and mean absolute/squared error and  $R^2$  score for the latter. Also dedicated scoring metrics for symbolic knowledge, as FiRe, may be used [26].

## 4. Experiments

Experiments to assess the capabilities of CReEPy applied to classification and regression tasks in comparison with state-of-the-art clustering and other ML and SKE techniques are reported in the following. The adopted ExACT and CReEPy implementations are included in the PSyKE framework<sup>2</sup> [27, 28, 29, 30].

### 4.1. Predictive Performance and Readability Assessments

To assess the capabilities of CReEPy in explaining opaque ML predictors we carried out several experiments involving real-world data sets. We selected the Iris data set<sup>3</sup> [31] as a case study for classification and 6 data sets from real use cases taken from the StairwAI EU Project<sup>4</sup> as case studies for regression.

#### 4.1.1. Classification: The Iris Data Set Case Study

The Iris data set represents a simple classification task with 4 continuous input features expressing as many characteristics of iris flowers. The target is the species of the flowers, which in this specific context may assume 3 possible distinct values. The data set is reported in Figure 1a. Only the 2 most relevant features are shown, i.e., petal length and width expressed in cm.

Our experiments on this data set are based on a  $k$ -nearest neighbours ( $k$ -NN) opaque predictor parametrised with  $k = 7$ . The corresponding decision boundaries are reported in Figure 1b.

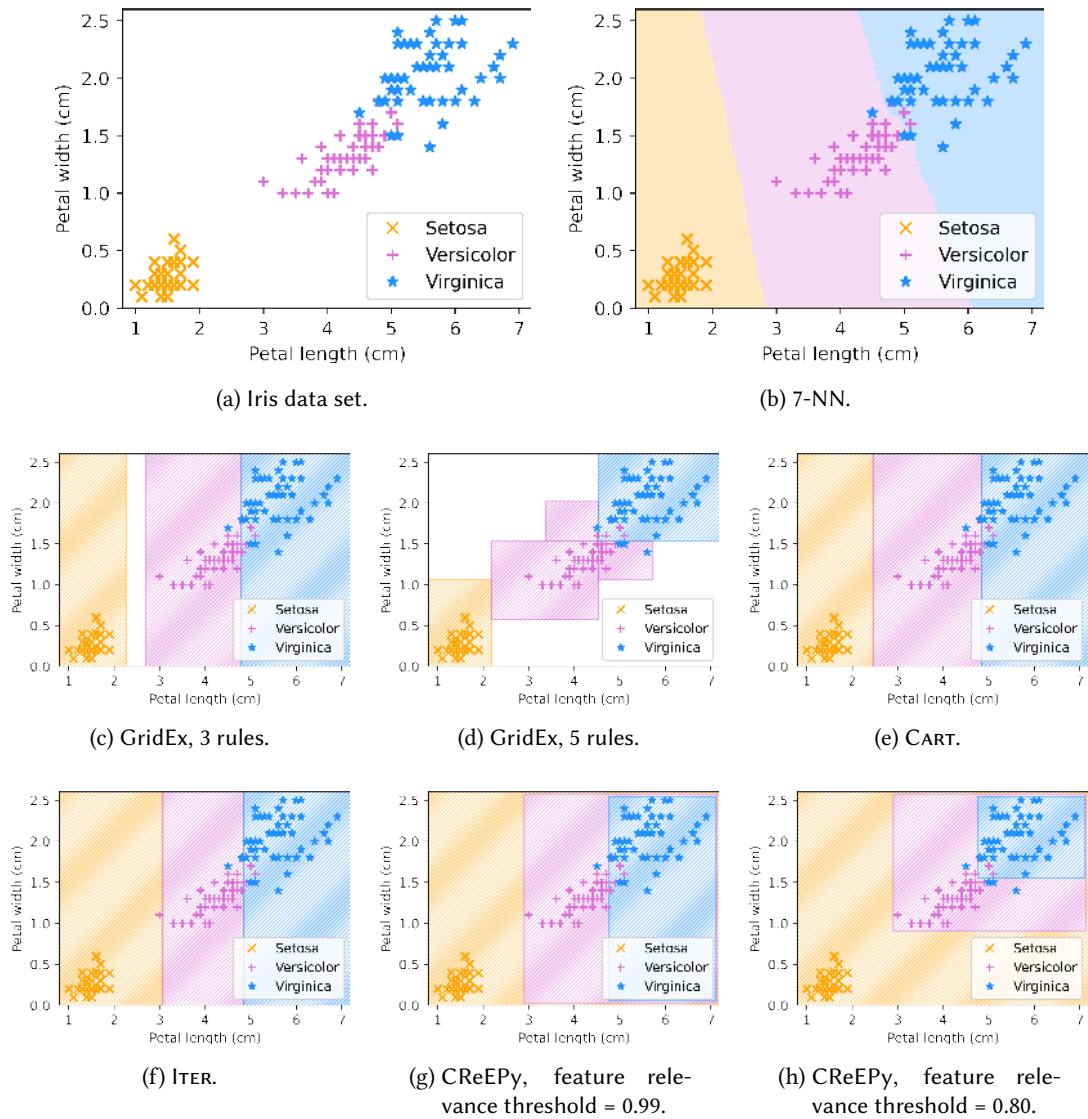
Decision boundaries identified by CReEPy and other SKE techniques are reported in the other panels of Figure 1. The parameters used for the SKE techniques are reported in the following list.

**GridEx** We adopted 2 different instances of GridEx. The one reported in Figure 1c produces 3 output rules (one per possible output class) and performs 14 slices only along input features having importance greater than 0.99—i.e., only along the most important feature,

<sup>2</sup>Code available at <https://github.com/psykei/psyke-python>

<sup>3</sup><https://archive.ics.uci.edu/ml/datasets/iris>

<sup>4</sup><https://cordis.europa.eu/project/id/101017142>; data sets are publicly available at <https://zenodo.org/record/5838437>



**Figure 1:** Symbolic knowledge extraction performed on the Iris data set.

the petal length. Conversely, the GridEx instance shown in Figure 1d performs 5 slices along each input dimension having importance greater than 0.80—i.e., petal length and width. This results in the 5 output rules depicted in the figure.

**CART** The decision boundaries reported in Figure 1e are obtained by growing an unbounded decision tree (no constraints on the tree depth, nor on the leaf amount).

**ITER** The ITER instance producing the input space partitioning reported in Figure 1f has been parametrised with a minimum cube update of 0.15 and an error threshold of 0.1. The



---

**Listing 1** Rules extracted with CReEPy for the Iris data set. Feature relevance threshold = 0.99.

---

```
iris(PetalLength, PetalWidth, SepalLength, SepalWidth, virginica) :-  
    PetalLength in [4.75, 6.90].  
iris(PetalLength, PetalWidth, SepalLength, SepalWidth, versicolor) :-  
    PetalLength in [2.90, 6.90].  
iris(PetalLength, PetalWidth, SepalLength, SepalWidth, setosa) :-  
    PetalLength in [1.10, 6.90].
```

---

**Listing 2** Rules extracted with CReEPy for the Iris data set. Feature relevance threshold = 0.80.

---

```
iris(PetalLength, PetalWidth, SepalLength, SepalWidth, virginica) :-  
    PetalLength in [4.75, 6.90], PetalWidth in [1.55, 2.60].  
iris(PetalLength, PetalWidth, SepalLength, SepalWidth, versicolor) :-  
    PetalLength in [2.90, 6.90], PetalWidth in [0.90, 2.60].  
iris(PetalLength, PetalWidth, SepalLength, SepalWidth, setosa) :-  
    PetalLength in [1.10, 6.90], PetalWidth in [0.00, 2.60].
```

---

maximum number of allowed iterations and the minimum amount of samples to consider in each cube have been fixed to 600 and 150, respectively. The algorithm started from a single random cube.

**CReEPy** Figures 1g and 1h correspond to CReEPy instances with feature relevance thresholds equal to 0.99 and 0.80, respectively. The former implies to consider only the most relevant feature when performing the knowledge extraction. By relaxing the threshold to 0.80 the second most important input feature is considered, as for GridEx. The input space partitioning reported in Figure 1g is equivalent to the Prolog theory shown in Listing 1. The Prolog theory corresponding to the decision boundaries reported in Figure 1h is shown in Listing 2.

Table 1 summarises the predictive performance measured for each SKE technique. The number of extracted rules is also reported as an index of the human-interpretability extent. From the results reported in the table, it is evident that CReEPy, compared to other state-of-the-art analogous techniques, is able to achieve comparable or slightly better predictive performance. As for the amount of extracted rules, CReEPy provides 3 rules, the optimum result given that it is applied to a classification task having 3 possible outcomes.

#### 4.1.2. Regression: The StairwAI EU Project Case Study

Thanks to the versatility of the underlying clustering constituting the core of CReEPy, it is possible to apply this latter to regression tasks as well. All the data sets used as case studies are composed of continuous features; 2 of them have 5 input features, and the remaining have 1 input feature. A different BB has been applied to each data set to draw predictions. A comparison between the knowledge extraction performed on the aforementioned data sets by CReEPy and other state-of-the-art analogous methods (namely, GridEx, GridREx and CART) has been reported in Table 3. Each measurement has been averaged on 5 different executions run under analogous conditions. Results provided by different executions are almost identical or very close, so we omitted the results' standard deviation in the table. For each data set, the



Model	Extracted rules	Predictive performance	
		F <sub>1</sub> score (data)	F <sub>1</sub> score (BB)
7-NN	–	0.95	
GridEx	3	0.97	0.94
GridEx	5	0.94	0.92
CART	3	0.95	0.97
ITER	3	0.94	0.97
CReEPy, feature relevance threshold = 0.99	3	0.95	0.97
CReEPy, feature relevance threshold = 0.80	3	0.94	0.96

**Table 1**

Assessments for the SKE techniques applied on a 7-NN performing classification on the Iris data set.

ID	Name	Input variables	Output variables	BB MAE
#1	Anticipate	5	cost	0.4
#2	Anticipate	1	memory	4.1
#3	Anticipate	1	time	8.3
#4	Contingency	5	cost	1.5
#5	Contingency	1	memory	3.6
#6	Contingency	1	time	0.8

**Table 2**

Regression data sets used to test CReEPy and compare it to analogous techniques. For each data set are reported: an unique identifier, the name of the data set, the amount of considered input features, the name of the considered output feature, and the mean absolute error measured for the BB trained on the data set.

number of input variables and the mean absolute error (MAE) of the corresponding BB model are reported. For each extractor, the number of output rules (R), the mean absolute error w.r.t. the actual data (D), and the BB predictions are reported. For all these experiments we chose local linear combinations of the input variables as outputs for the regions approximated by the underlying ExACT instances. Indeed, the adoption of constant outputs resulted in more concise output rules having, however, far worse predictive performance.

It is important to note that only the mean absolute error is reported as a measure of predictive performance even though other metrics are available, such as the mean squared error or the R<sup>2</sup> score. The number of extracted rules is taken as a readability measure since readability for humans decreases if the amount of rules increases. Another index used to assess and compare the quality of extractors is the completeness of the extracted knowledge [32], but in this particular case study is not relevant, since all the procedures achieve a level of completeness above 99%.

CReEPy proved to be superior to CART from a predictive performance perspective since local linear combinations of input variables better approximate the data set/BB outputs than constant values. Furthermore, a readability comparison between the two extractors shows that CReEPy is able to halve the extracted rule amount in 50% of experiments. Analogous considerations hold for the comparison with GridEx, with even a more evident readability enhancement when

Data set	CReEPy MAE			GridREx MAE			GridEx MAE			CART MAE		
	R	D	BB	R	D	BB	R	D	BB	R	D	BB
#1	3	1.5	1.5	5	1.9	2.0	5	14.6	14.6	4	14.7	14.7
#2	2	4.9	3.3	5	4.9	3.2	5	15.0	14.7	4	17.4	17.0
#3	3	11.5	7.9	4	11.1	6.6	5	17.7	15.0	4	16.7	12.9
#4	4	26.6	26.8	4	24.4	24.6	5	28.5	28.6	4	25.1	25.1
#5	2	4.7	2.3	4	4.5	2.3	4	4.7	2.3	4	4.5	2.3
#6	2	1.0	0.7	5	1.0	0.7	5	3.1	3.1	4	3.9	3.8

**Table 3**

Results of CReEPy applied to the 6 data sets described in Table 2. For each data set the number of extracted rules (R) and the MAE w.r.t. the data (D) and the underlying BB model are provided. Results are compared with those of GridREx, GridEx, and CART applied to the same data sets.

considering CReEPy.

The most interesting comparison is with GridREx, able as well to provide local approximations in the form of linear input variable combinations. By exploiting CReEPy it is possible to achieve approximately the same predictive performance shown by GridREx with far better readability (for instance, 2 output rules instead of 5 or 4, by considering experiments on data sets #2, #5 and #6). In conclusion, our proposed knowledge extractor performing an upstream interpretable clustering via EXACT is absolutely competitive with state-of-the-art SKE algorithms.

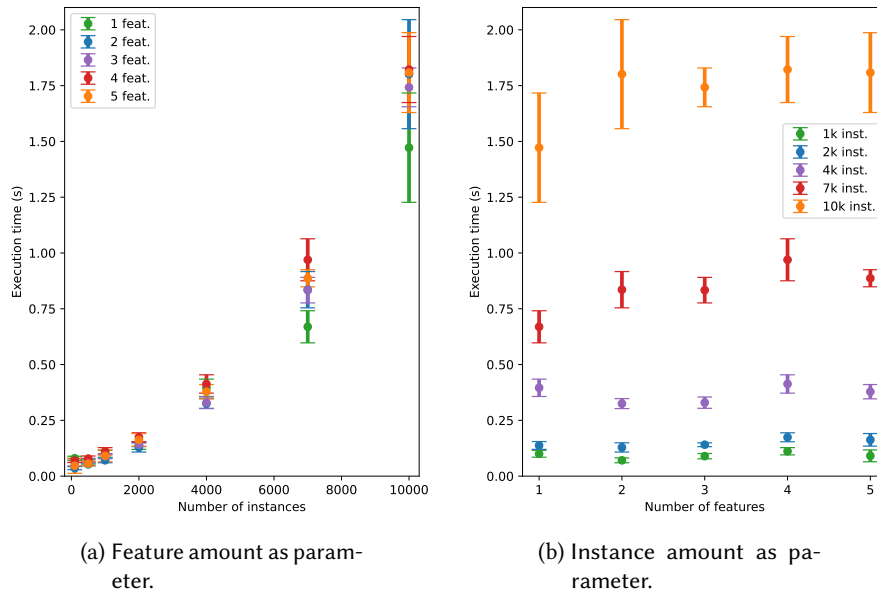
## 4.2. Computational Time Assessments

Our experiments are completed by a quantitative assessment of the computational time required by CReEPy to perform the knowledge extraction. Tests consider data set #1, by performing both row and column slicing on it. In particular, a comparison of the computational time required to handle the data set with different amounts of input features and instances has been performed. Results are reported in Figure 2. Measurements have been averaged upon 100 executions.

From Figure 2a it is clear that the execution time grows by augmenting the number of training instances. Clues on the independence of required time w.r.t. the number of input features may be found in the same figure. Such independence is clearly noticeable in Figure 2b, showing that the computational time is always smaller than 2, 1 and 0.5 seconds for 10 000, 7 000 and 4 000 instances, respectively, regardless of the input feature amount. In conclusion, we suggest fastening CReEPy, when necessary, by reducing the amount of training data points instead of the number of input features.

## 5. Conclusions

In this paper we present a SKE technique named CReEPy, applicable to any kind of opaque ML classifier or regressor working upon data sets described by continuous input features. CReEPy is able to outperform existing techniques from both the predictive performance and human-readability perspectives. CReEPy is a two-phase algorithm since it performs an explainable



**Figure 2:** Execution time of CReEPy w.r.t. the number of input features of the domain and the number of instances adopted for the training.

clustering technique on the training data before the proper knowledge-extraction phase. The human readability of the extracted knowledge is ensured since it is provided to users in the form of a logic theory adhering to the Prolog syntax.

The upstream clustering techniques designed for CReEPy and also described here are ExACT and CREAM. These algorithms take advantage of GMMs and DBSCAN to detect clusters and approximate them with human-interpretable hypercubic regions described in terms of interval inclusion constraints on the input features. ExACT and CREAM may as well be used as a stand-alone explainable clustering procedure to perform clustering other than classification and regression.

Our future works will be focused on enhancing the rationale behind the ExACT’s and CREAM’s region approximation and possibly on the adoption of deep clustering techniques instead of GMMs and DBSCAN, as in the current versions. On the other hand, CReEPy may benefit from an automatic technique enabling parameter auto-tuning and in particular we plan to implement a procedure aimed at highlighting the best values for the maximum depth and the predictive error threshold parameters.

## Acknowledgments

This work has been supported by the EU ICT-48 2020 project TAILOR (No. 952215).

## References

- [1] European Commission, C. Directorate-General for Communications Networks, Technology, Ethics guidelines for trustworthy AI, Publications Office, 2019. doi:doi/10.2759/346720.
- [2] European Commission, AI Act – Proposal for a regulation of the european parliament and the council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain union legislative acts, <https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52021PC0206>, 2021.
- [3] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, D. Pedreschi, A survey of methods for explaining black box models, *ACM Computing Surveys* 51 (2018) 1–42. doi:10.1145/3236009.
- [4] S. Ayache, R. Eyraud, N. Goudian, Explaining black boxes on sequential data using weighted automata, in: *International Conference on Grammatical Inference*, PMLR, 2019, pp. 81–103.
- [5] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature Machine Intelligence* 1 (2019) 206–215. doi:10.1038/s42256-019-0048-x.
- [6] E. M. Kenny, C. Ford, M. Quinn, M. T. Keane, Explaining black-box classifiers using post-hoc explanations-by-example: The effect of explanations and error-rates in XAI user studies, *Artificial Intelligence* 294 (2021) 103459. doi:10.1016/j.artint.2021.103459.
- [7] R. Calegari, G. Ciatto, A. Omicini, On the integration of symbolic and sub-symbolic techniques for XAI: A survey, *Intelligenza Artificiale* 14 (2020) 7–32. doi:10.3233/IA-190036.
- [8] G. Bologna, C. Pellegrini, Three medical examples in neural network rule extraction, *Physica Medica* 13 (1997) 183–187. URL: <https://archive-ouverte.unige.ch/unige:121360>.
- [9] Y. Hayashi, R. Setiono, K. Yoshida, A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders, *Artificial intelligence in Medicine* 20 (2000) 205–216. doi:10.1016/s0933-3657(00)00064-6.
- [10] B. Baesens, R. Setiono, C. Mues, J. Vanthienen, Using neural network rule extraction and decision tables for credit-risk evaluation, *Management Science* 49 (2003) 312–329. doi:10.1287/mnsc.49.3.312.12739.
- [11] A. Hofmann, C. Schmitz, B. Sick, Rule extraction from neural networks for intrusion detection in computer networks, in: *2003 IEEE International Conference on Systems, Man and Cybernetics*, volume 2, IEEE, 2003, pp. 1259–1265. doi:10.1109/ICSMC.2003.1244584.
- [12] M. T. A. Steiner, P. J. Steiner Neto, N. Y. Soma, T. Shimizu, J. C. Nievola, Using neural network rule extraction for credit-risk evaluation, *International Journal of Computer Science and Network Security* 6 (2006) 6–16. URL: [http://paper.ijcsns.org/07\\_book/200605/200605A02.pdf](http://paper.ijcsns.org/07_book/200605/200605A02.pdf).
- [13] L. Franco, J. L. Subirats, I. Molina, E. Alba, J. M. Jerez, Early breast cancer prognosis prediction and rule extraction using a new constructive neural network algorithm, in: *Computational and Ambient Intelligence (IWANN 2007)*, volume 4507 of *LNCS*, Springer, 2007, pp. 1004–1011. doi:0.1007/978-3-540-73007-1\_121.
- [14] F. Sabbatini, C. Grimani, Symbolic knowledge extraction from opaque predictors applied to cosmic-ray data gathered with LISA Pathfinder, *Aeronautics and Aerospace Open Access*

Journal 6 (2022) 90–95. URL: <https://doi.org/10.15406/aaobj.2022.06.00145>. doi:10.15406/aaobj.2022.06.00145.

- [15] R. Andrews, J. Diederich, A. B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, *Knowledge-Based Systems* 8 (1995) 373–389. doi:10.1016/0950-7051(96)81920-4.
- [16] R. Setiono, W. K. Leow, J. M. Zurada, Extraction of rules from artificial neural networks for nonlinear regression, *IEEE Transactions on Neural Networks* 13 (2002) 564–577. doi:10.1109/TNN.2002.1000125.
- [17] J. Huysmans, B. Baesens, J. Vanthienen, ITER: An algorithm for predictive regression rule extraction, in: *Data Warehousing and Knowledge Discovery (DaWaK 2006)*, Springer, 2006, pp. 270–279. doi:10.1007/11823728\_26.
- [18] F. Sabbatini, G. Ciatto, A. Omicini, GridEx: An algorithm for knowledge extraction from black-box regressors, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främmling (Eds.), *Explainable and Transparent AI and Multi-Agent Systems. Third International Workshop, EXTRAAMAS 2021, Virtual Event, May 3–7, 2021, Revised Selected Papers*, volume 12688 of *LNCS*, Springer Nature, Basel, Switzerland, 2021, pp. 18–38. doi:10.1007/978-3-030-82017-6\_2.
- [19] F. Sabbatini, G. Ciatto, R. Calegari, A. Omicini, Hypercube-based methods for symbolic knowledge extraction: Towards a unified model, in: A. Ferrando, V. Mascardi (Eds.), *WOA 2022 – 23rd Workshop “From Objects to Agents”*, volume 3261 of *CEUR Workshop Proceedings*, Sun SITE Central Europe, RWTH Aachen University, 2022, pp. 48–60. URL: <http://ceur-ws.org/Vol-3261/paper4.pdf>.
- [20] F. Sabbatini, G. Ciatto, R. Calegari, A. Omicini, Towards a unified model for symbolic knowledge extraction with hypercube-based methods, *Intelligenza Artificiale* 17 (2023) 63–75. URL: <https://doi.org/10.3233/IA-230001>. doi:10.3233/IA-230001.
- [21] F. Sabbatini, R. Calegari, Symbolic knowledge extraction from opaque machine learning predictors: GridREx & PEDRO, in: G. Kern-Isberner, G. Lakemeyer, T. Meyer (Eds.), *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022*, 2022. URL: <https://proceedings.kr.org/2022/57/>. doi:10.24963/kr.2022/57.
- [22] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, *Classification and Regression Trees*, CRC Press, 1984.
- [23] F. Sabbatini, R. Calegari, ExACT explainable clustering: Unravelling the intricacies of cluster formation, in: *Proceedings of the 2nd International Workshop on Knowledge Diversity, KoDis 2023, Rhodes, Greece, September 2–8, 2023 (to appear)*, 2023.
- [24] F. Sabbatini, R. Calegari, Explainable Clustering with CREAM, in: *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning*, 2023, pp. 593–603. URL: <https://doi.org/10.24963/kr.2023/58>. doi:10.24963/kr.2023/58.
- [25] F. Sabbatini, R. Calegari, Bottom-up and top-down workflows for hypercube- and clustering-based knowledge extractors, in: D. Calvaresi, A. Najjar, A. Omicini, R. Aydogan, R. Carli, G. Ciatto, K. Främmling (Eds.), *Explainable and Transparent AI and Multi-Agent Systems. Fifth International Workshop, EXTRAAMAS 2023, London, UK, May 29, 2023, Revised Selected Papers*, volume 14127 of *LNCS*, Springer Cham, Basel, Switzerland, 2023, pp. 116–129. doi:10.1007/978-3-031-40878-6\_7.

- [26] F. Sabbatini, R. Calegari, Symbolic knowledge-extraction evaluation metrics: The FiRe score, in: K. Gal, A. Nowé, G. J. Nalepa, R. Fairstein, R. Rădulescu (Eds.), Proceedings of the 26th European Conference on Artificial Intelligence, ECAI 2023, Kraków, Poland. September 30 – October 4, 2023, 2023. URL: <https://ebooks.iospress.nl/doi/10.3233/FAIA230496>. doi:10.3233/FAIA230496.
- [27] F. Sabbatini, G. Ciatto, R. Calegari, A. Omicini, On the design of PSyKE: A platform for symbolic knowledge extraction, in: R. Calegari, G. Ciatto, E. Denti, A. Omicini, G. Sartor (Eds.), WOA 2021 – 22nd Workshop “From Objects to Agents”, volume 2963 of *CEUR Workshop Proceedings*, Sun SITE Central Europe, RWTH Aachen University, 2021, pp. 29–48. 22nd Workshop “From Objects to Agents” (WOA 2021), Bologna, Italy, 1–3 September 2021. Proceedings.
- [28] F. Sabbatini, G. Ciatto, R. Calegari, A. Omicini, Symbolic knowledge extraction from opaque ML predictors in PSyKE: Platform design & experiments, *Intelligenza Artificiale* 16 (2022) 27–48. URL: <https://doi.org/10.3233/IA-210120>. doi:10.3233/IA-210120.
- [29] F. Sabbatini, G. Ciatto, A. Omicini, Semantic Web-based interoperability for intelligent agents with PSyKE, in: D. Calvaresi, A. Najjar, M. Winikoff, K. Främling (Eds.), *Explainable and Transparent AI and Multi-Agent Systems*, volume 13283 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 124–142. URL: [http://link.springer.com/10.1007/978-3-031-15565-9\\_8](http://link.springer.com/10.1007/978-3-031-15565-9_8). doi:10.1007/978-3-031-15565-9\_8.
- [30] R. Calegari, F. Sabbatini, The PSyKE technology for trustworthy artificial intelligence 13796 (2023) 3–16. URL: [https://doi.org/10.1007/978-3-031-27181-6\\_1](https://doi.org/10.1007/978-3-031-27181-6_1). doi:10.1007/978-3-031-27181-6\_1, xXI International Conference of the Italian Association for Artificial Intelligence, AIxIA 2022, Udine, Italy, November 28 – December 2, 2022, Proceedings.
- [31] R. A. Fisher, The use of multiple measurements in taxonomic problems, *Annals of Eugenics* 7 (1936) 179–188. doi:10.1111/j.1469-1809.1936.tb02137.x.
- [32] F. Sabbatini, R. Calegari, On the evaluation of the symbolic knowledge extracted from black boxes, in: *AAAI 2023 Spring Symposium Series* (to appear), San Francisco, California, 2023.