Bottom-Up and Top-Down Workflows for Hypercube- And Clustering-Based Knowledge Extractors

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. Machine learning opaque models, currently exploited to carry out a wide variety of supervised and unsupervised learning tasks, are able to achieve impressive predictive performances. However, they act as black boxes (BBs) from the human standpoint, so they cannot be entirely trusted in critical applications unless there exists a method to extract symbolic and human-readable knowledge out of them.

In this paper we analyse a recurrent design adopted by symbolic knowledge extractors for BB predictors—that is, the creation of rules associated with hypercubic input space regions. We argue that this kind of partitioning may lead to suboptimum solutions when the data set at hand is sparse, high-dimensional, or does not satisfy symmetric constraints. We then propose two different knowledge-extraction workflows involving clustering approaches, highlighting the possibility to outperform existing knowledge-extraction techniques in terms of predictive performance on data sets of any kind.

Keywords: explainable AI · symbolic knowledge extraction · clustering

1 Introduction

Machine learning (ML) models in general – and (deep) artificial neural networks in particular – are nowadays exploited to draw predictions in almost every application area [26]. However, when facing *critical* domains – e.g., involving human health, wealth, or freedom – ML models behaving as opaque predictors are not an acceptable choice. The *opaque* nature of these models makes them unintelligible for humans and this is the reason why they are called *black boxes* (BBs). Nonetheless, explainability can be obtained from BBs via several strategies [17]. For instance, one can rely uniquely on *interpretable* models [27], or build explanations by applying reverse engineering to the BB behaviour [21]. The former option is not always practicable, since interpretable models as linear regressors and shallow decision trees are not always prediction-effective as more complex models—for instance, random forests or deep artificial neural networks. On the other hand, the latter approach allows users to combine the impressive predictive capabilities of opaque predictors with the human readability proper of symbolic models.

The majority of present literature offers a wide variety of procedures explicitly designed to extract symbolic knowledge from opaque ML classifiers [2,13,14,23,24,40,43, for instance]. A smaller set of procedures is dedicated to BB regressors [20,28,32,35,38,39,41, to cite some], whereas few exceptions are able to consider both categories [6,7,9,11,22,36]. A large amount of available techniques depend on the existence of software libraries supporting ML in general (e.g., Python's Scikit-Learn[1] [25]) and symbolic knowledge extraction in particular (for instance, PSyKE[2] [10,29,31,33]).

Unfortunately, any method offers peculiar advantages, but at the same time, it is subject to drawbacks and limitations. In the following we focus on the issues deriving from the extraction of a particular kind of rules from BB classifiers and regressors. More in detail, we observe that opaque predictors in general – and regressors in particular – are often explained via a human-interpretable partitioning of the input space. A typical design choice is to identify hypercubic regions of the input feature space enclosing similar instances and then to associate symbolic knowledge to each region, for instance in the form of *first-order logic* rules [20,28,32]. We agree that rules associated with hypercubic regions are the best choice from the human-readability perspective since they enable the description of an input space region in terms of constraints on single dimensions (e.g., $0.3 < X < 0.6, 0.5 < Y < 0.75$ for a hypercube in a 2-dimensional space having features X and Y). However, this solution may lead to the creation of suboptimum clusters in several scenarios, for instance, if the partitioning into hypercubes is performed following some sort of top-down symmetric procedure on asymmetric data sets, or if the partitioning is bottom-up and performed on sparse data sets. In this paper we suggest exploiting clustering techniques on the data set used to train the BB before extracting knowledge from it, in order to preemptively find and distinguish relevant input regions with the corresponding boundaries. This theoretically allows extractors to: *(i)* automatically tune the number of output rules w.r.t. the number of relevant regions found; *(ii)* give priority to more relevant regions, for instance those containing more input instances or having the largest volume; *(iii)* avoid unsupervised partitioning of the input feature space, otherwise leading to suboptimum solutions in terms of readability and/or fidelity. Given that our proposed workflows are based on the training data sample distribution and require no knowledge of the adopted opaque predictor, it may be possible to build pedagogical knowledge-extraction algorithms adhering to them.

[1] https://scikit-learn.org/stable/index.html.
[2] https://github.com/psykei/psyke-python.

Accordingly, in Sect. 2 an overview of symbolic knowledge extraction in general and some techniques in particular is provided. In Sect. 3 the main drawbacks of existing knowledge-extraction algorithms based on hypercubes are highlighted and discussed. We then propose novel clustering-based workflows for knowledge extraction in Sect. 4 and our work is finally summarised in Sect. 5.

2 Related Works

A predictive model can be defined as *interpretable* if human users are able to easily understand its behaviour and outputs [12]. Since the majority of modern ML predictors store the knowledge acquired during their training phase in a *sub-symbolic* way, they behave and appear to the human perspective as unintelligible black boxes. The explainable artificial intelligence community has proposed a variety of methods to enrich BB predictions with corresponding interpretations/explanations without renouncing their superior predictive performance. Usually, the proposed methods consist of creating an interpretable, mimicking model by inspecting the underlying BB in terms of internal behaviour and/or input/output relationships. For instance, RefAnn analyses the architecture of neural network regressors with one hidden layer to obtain information about the internal parameters and thus build human-readable *if-then* rules having a linear combination of the input features as postconditions [39]. This kind of technique is called *decompositional*. On the other hand, when the internal structure of the BB is not considered to build explanations, algorithms are classified as *pedagogical*.

Symbolic knowledge-extraction techniques are currently applied in several critical areas, such as healthcare, credit-risk evaluation, intrusion detection systems, and many others [3–5,8,16,18,19,34,37,42].

2.1 ITER

The Iter algorithm [20] is a pedagogical technique to extract symbolic knowledge from BB regressors. Iter induces a hypercubic partitioning of the input feature space following a bottom-up strategy, starting with points in the multidimensional space and expanding them until the final hypercubic output regions. According to the Iter design, all the produced hypercubes are non-overlapping and they do not exceed the input feature space.

After the hypercube expansion Iter associates an *if-then* rule to each cube, selecting as a postcondition the mean output value of all the instances contained in the cube. This behaviour may be relaxed to support classification tasks or regression tasks having outputs described through linear laws by adopting the generalisation proposed in [30].

The algorithm's main advantage is to be capable of constructing hypercubes having different sides' lengths. However, especially when dealing with high-dimensional data sets, it may present several criticalities related to the hypercubes' expansion.

2.2 GridEx and GridREx

The GridEx algorithm [32] is a different pedagogical technique to extract symbolic knowledge from BB regressors. It is applicable under the same conditions of Iter and outputs the same kind of knowledge, but they differ in the strategy adopted during the input space partitioning. GridEx is not a bottom-up algorithm; conversely, it follows a top-down strategy, starting from the whole input space and recursively partitioning it into smaller hypercubic regions, according to a user-defined threshold acting as a trade-off criterion between readability (in terms of the number of extracted rules) and fidelity of the output model (intended as its ability to mimic the underlying BB).

Also for GridEx it is possible to adopt a generalisation enabling its application for classification tasks [30]. On the other hand, the GridREx algorithm [28] is the extension of GridEx providing linear combinations of the input features as outputs associated with the identified hypercubic relevant regions.

Amongst the common advantages of GridEx and GridREx there is the ability to automatically refine the output regions according to the provided threshold, as well as to perform a merging step after each split, when possible. In particular, the merging step consists of the pairwise unification of adjacent hypercubes to reduce the number of output rules (to enhance human readability) and it is based on the similarity between the samples included in each cube (to avoid a predictive performance worsening). It is useful since the splitting phase may create excessive amounts of disjoint but adjacent, similar regions.

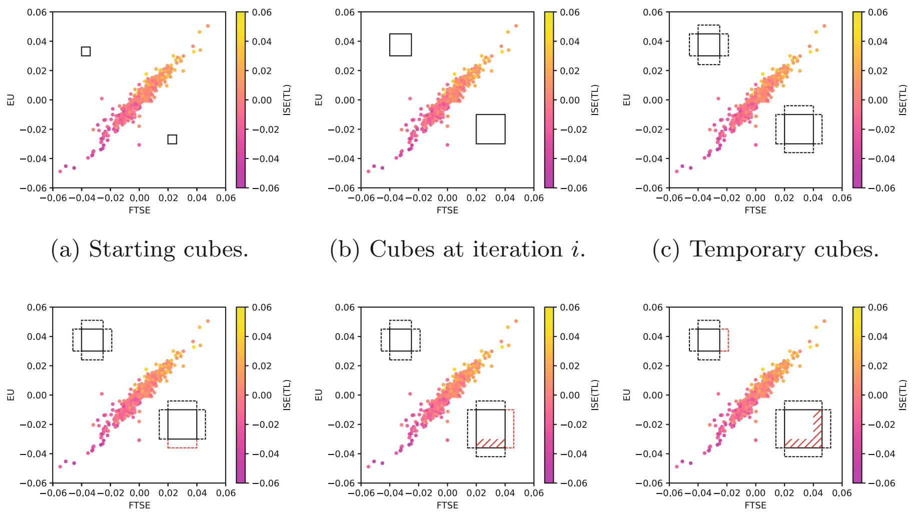
3 Limitations of Existing Knowledge-Extraction Methods

In order to point out the main limitations affecting existing symbolic knowledge-extraction techniques based on hypercubic partitioning a clear insight into how the partitioning is performed has fundamental importance. For this reason the behaviour of Iter, GridEx and GridREx is further detailed in the following with the aim of spotting their major drawbacks and providing possible solutions. Examples of these knowledge-extraction algorithms applied to real-world data sets are also reported to strengthen our argumentation. Iter's examples are considered on the Istanbul Stock Exchange data set[3] [1], describing a regression task with 7 continuous input features plus another input feature representing a timestamp. Examples for GridEx are based on the Wine Quality data set[4] [15], composed of 13 continuous input features.

3.1 ITER

The Iter procedure is exemplified for the Istanbul Stock Exchange data set in Fig. 1, where only the 2 most relevant input features are reported, i.e., the stock market return index of UK (FTSE) and the MSCI European index (EU). The

[3] https://archive.ics.uci.edu/ml/datasets/ISTANBUL+STOCK+EXCHANGE.
[4] https://archive.ics.uci.edu/ml/datasets/wine.



(a) Starting cubes. (b) Cubes at iteration i. (c) Temporary cubes.

(d) Best temporary cube. (e) Merge (iteration i). (f) Merge (iteration $i + 1$).

Fig. 1. Example of ITER hypercube expansion.

figure is a mere sketch depicting a possible undesired execution scenario of the algorithm, starting from random points belonging to negligible input regions.

During the execution of ITER a certain number of infinitesimally small hypercubes (i.e., multidimensional points) are created within the input feature space (cf. Fig. 1a) and they are iteratively expanded until some stopping criteria are met – i.e., the whole space is covered or it is not possible to expand the existing cubes nor to create new ones –, or after a specified amount of iterations.

The expansion follows this strategy, for a generic iteration i (cf. Fig. 1b):

1. build adjacent temporary cubes around the existing ones (2 temporary cubes per input dimension per existing cube; cf. Fig. 1c, temporary cubes are represented with dashed perimeter);
2. select the best temporary cube—i.e., the most similar to the corresponding adjacent existing cube (cf. Fig. 1d, the best temporary cube is highlighted with red perimeter);
3. merge the best temporary cube with the existing one (cf. Fig. 1e, the merged region is highlighted with red hatches);
4. repeat from step 1 for every successive iteration (cf. Fig. 1f).

The temporary cube selected to be merged may become instead a new independent cube if the similarity w.r.t. its adjacent cube is below a user-defined threshold. The number of starting cubes as well as the size of the temporary cubes and the maximum number of allowed iterations are other hyper-parameters to be provided by users. Conversely, the position of the initial cubes is randomly chosen, so it represents a source of non-determinism.

It is important to notice that at each algorithm iteration, only one hypercube amongst all the available temporary cubes is merged. The others are discarded and the majority of them are created again without modifications during the successive iteration. This may lead to an enormous waste of computational time and resources due to the repetition of (the same) useless calculations, other than the possibility to exceed the maximum allowed iterations without having convergence. The absence of convergence results in a non-exhaustive partitioning of the input feature space, which in turn implies the inability to predict output values of data samples belonging to regions that are not covered by a hypercube (i.e., there are no human-interpretable predictive rules associated with uncovered regions). Conversely, the coverage of the whole input feature space enables drawing predictions for any input instance.

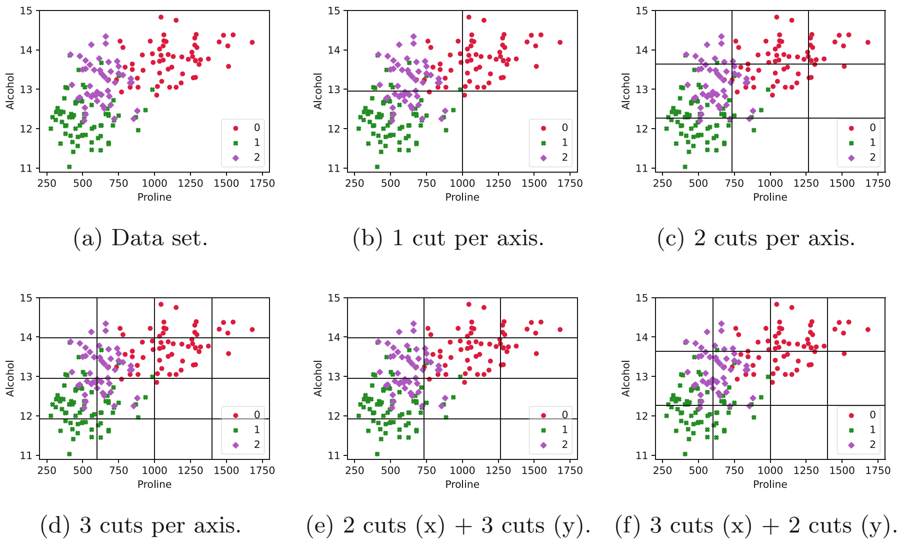
Given the existence of the non-exhaustivity issue, it is of paramount importance the ability to focus on relevant input space subregions first, actually missing in the ITER design, since cubes are initialised randomly and expanded regardless of the subregion relevance. A naive notion of relevance for an input space subregion may be the amount of contained data set samples. If a training data set is representative of a certain task, it is admissible to envisage that if an input space subregion has no training instances, then probably there will not be data samples belonging to that same region to be predicted in the future, so the region is negligible, or at least not compelling. We highlight that it is not sufficient to build the starting cubes around existing training samples randomly chosen, since these may still be outliers. Conversely, the notion of region relevance should be intertwined with that of instance density and therefore the most relevant regions should be those containing more training instances. Since ITER is actually unaware of the sample density, we believe that this procedure could be improved by making the hypercube initialisation and expansion density-driven, or at least density-aware, without neglecting the pivotal similarity criterion of the original design.

3.2 GridEx and GridREx

The GridEx algorithm applied to the Wine Quality data set is visually shown in Fig. 2 to highlight its weaknesses. Only 2 out of 13 input features are reported in the figure to avoid a chaotic representation, i.e., the most relevant for the classification task.

GridEx considers the data point distribution during its execution, intended as the location of the data points inside the input feature space, but it neglects the output value of the training instances during the assignment of a priority to input feature space subregions. Indeed, it identifies 3 kinds of subregions:

negligible regions i.e., those without training instances belonging to them. These regions may be neglected without a noticeable impact on the overall extractor performance since the probability of existing instances enclosed by them is very low;



(a) Data set. (b) 1 cut per axis. (c) 2 cuts per axis.

(d) 3 cuts per axis. (e) 2 cuts (x) + 3 cuts (y). (f) 3 cuts (x) + 2 cuts (y).

Fig. 2. Example of GridEx hypercubic partitioning.

permament regions i.e., those containing training samples and for which the estimated predictive error is below the user-defined threshold. These regions have a satisfying predictive performance from the user standpoint and require no further partitioning;

eligible regions i.e., those containing training data but having an associated predictive error beyond the user-defined threshold. These regions require to be refined with further splitting since they hinder the overall predictive performance of the model.

By following this strategy the complete coverage of the input feature space is not granted. Nonetheless, very high coverage rates are achieved when predicting new unknown instances.

For the examples reported in Fig. 2 only actual GridEx instances performing a single iteration are considered. This means that the eligible regions do have not the possibility to be further refined. By observing Figs. 2b to 2f it is possible to notice that the cuts performed by GridEx are perpendicular to the axes and create for each dimension a set of partitions having the same size, even if the number of cuts may differ for each dimension (cf. Figs. 2e and 2f). To apply a different amount of cuts for each input dimension it is necessary to adopt the adaptive version of GridEx, selecting a number of slices that are proportional to the dimensions' relevance.

The symmetric strategy is the main disadvantage of GridEx, in its base and adaptive versions, and the same occurs with GridREx, since it follows the same hypercubic partitioning strategy of GridEx. This design choice may lead to a drop in the predictive performance if the identified hypercubes include portions of separated input regions. Given the symmetric nature of the algorithm, there is no certainty that the predictive error measured for the hypercubic regions can be reduced by augmenting the number of partitions, since a more fine-grained partitioning may lead in any case to a poor approximation on asymmetric data sets. We believe that GridEx and GridREx may be improved by performing slices that are not blindly symmetric, but somehow aware of the training data points' outputs. More in detail, cuts should avoid splitting into different regions training samples that are similar, as well as avoid creating regions containing too different training instances. To achieve this goal clustering may be performed before the cuts in order to identify different clusters of data and therefore avoid cuts in the proximity of the clusters' centroids. Conversely, cuts in correspondence of the intersection of two clusters should be encouraged.

4 Clustering-Based Approaches

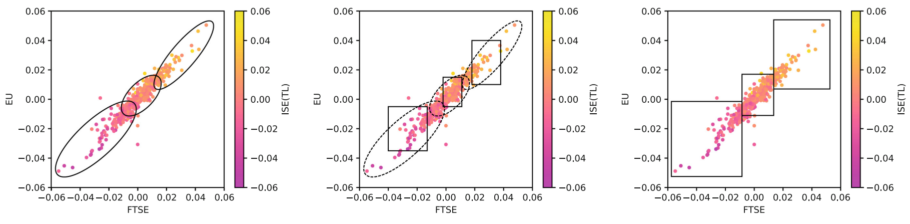
Density and similarity estimates inside the training set input space may be obtained, for instance, via the application of clustering techniques to the available data. With reference to Fig. 1, three clusters may be identified: one enclosing the data points in the bottom-left part of the plot, one for the middle samples and the other for the top-right instances. On the other hand, for Fig. 2 the clusters may be defined according to the output expected classes and therefore also in this case three distinct clusters are identified.

An ideal extraction technique should be able to identify relevant clusters of data in a *fast* and *flawless* way, especially if clusters are linearly separable. An ideal procedure should also be able to approximate these clusters according to a human-interpretable format, for instance with hypercubic regions, enclosing the training data points without overlaps between approximated regions. We stress that this is only a desideratum, since real-world data sets seldom contain linearly separable classes of instances, but the design of a clustering-based knowledge extractor should take this ideal goal into account.

Bottom-up strategies like the one adopted by ITER generally result in time-consuming executions in the n-dimensional domain, with large n. This may result in a very slow convergence or even incomplete input space coverage. It is possible to fasten the ITER convergence by acting on the algorithm parameters, but at the expense of a coarser partitioning. This latter inconvenience is the same encountered by using GridEx, since it induces an equally spaced grid not always able to capture the properties of the data distribution inside the input feature space. If a grid cell only contains instances belonging to a single cluster, the predictive error will be small. Otherwise, it will be less or more large depending on the amount of contamination of each grid cell.

An optimum, fast bottom-up hypercubic-based extraction technique can be obtained by performing the following steps:

1. apply a clustering technique to the data set and therefore identify the different relevant regions;



(a) Initial clustering. (b) Starting cubes. (c) Final partitioning.

Fig. 3. Example of an ideal bottom-up clustering- and hypercube-based symbolic knowledge extraction.

2. construct hypercubes to include the whole found regions, or a part of them, since this shape can be straightforwardly represented in a symbolic and human-readable format;
3. refine the hypercubes to enhance coverage and predictive performance of the explainable model, for instance by expanding the approximated regions;
4. remove the overlaps amongst the hypercubic regions, or impose an order to avoid ambiguity in evaluating the membership of instances to regions;
5. describe each hypercube in terms of the input features and then associate a corresponding human-interpretable output value to obtain the final explainable model.

The described workflow for an ideal bottom-up clustering-based extractor performing hypercubic partitioning of the input feature space is depicted in Fig. 3.

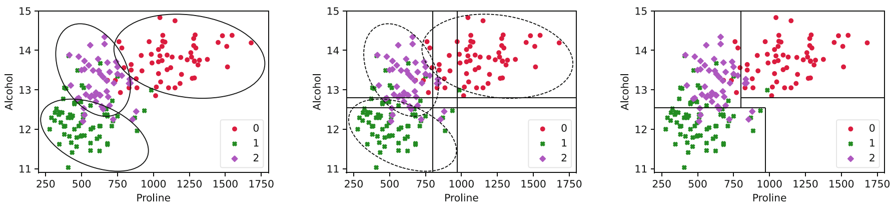
Conversely, an effective and fast top-down extraction technique based on hypercubic partitioning can be performed by substituting the middle steps of the workflow described above:

2. cut the input feature space in order to separate different clusters while avoiding spreading instances of a single cluster over multiple regions;
3. create hypercubic regions approximating the identified optimum cuts, avoiding overlapping cubes;
4. refine the hypercubes by recursively repeating the previous steps for each cube, to enhance the predictive performance of the explainable model.

The corresponding workflow for an ideal top-down knowledge-extraction procedure based on clustering is depicted in Fig. 4. Both Fig. 3 and Fig. 4 are conceptual sketches highlighting the fundamental steps of the aforementioned workflows, with the goal of guiding the future development of procedures adhering to the presented concepts.

4.1 Open Issues

Extraction techniques may surely benefit from cluster-aware partitioning methods. Accuracy in the selection of different clusters and in the construction



(a) Initial clustering. (b) Optimum cuts. (c) Final partitioning.

Fig. 4. Example of an ideal top-down clustering- and hypercube-based symbolic knowledge extraction.

of clustering-based hypercubes may enable the achievement of the following desiderata: extracting the minimum number of different predictive rules (one per cluster) having the lowest possible predictive error. However, a number of challenges arise from the aforementioned workflows. For instance:

i. How to select the correct number of clusters to identify, if unknown?
ii. How to handle outliers in the construction of hypercubes for the found regions and in deciding where to cut the input space?
iii. How to build hypercubes around clusters associated with overlapping hypercubic regions?
iv. How to discern amongst different clusters approximated by the same hypercubic region?

We believe that powerful strategies to describe non-trivial clusters may exploit *difference cubes* – e.g., regions of the input feature space having a non-cubic shape and described by the subtraction of cubic areas – and hierarchical clusters. We stress that the importance of adopting hypercubes to describe input regions depends on the possibility to define a hypercube in terms of single variables belonging to specific intervals, in a highly human-comprehensible form. This is not true when dealing with other representations—e.g., oblique rules, *M-of-N* rules.

5 Conclusions

In this paper we propose two clustering-based workflows to enhance the symbolic knowledge extraction from BB predictors in terms of computational complexity, fidelity, and predictive performance. The former is based on a bottom-up hypercubic approximation of the input feature space, resulting in a density-driven fast partitioning. Conversely, the latter is a top-down cutting of the input feature space providing hypercubic partitioning as well. Our methods can be exploited to build interpretable regions associated with human-readable logic rules based on an upstream clustering technique. In our future works we plan to implement and include in the PSyKE framework different knowledge extractors adhering to the presented concepts and capable of handling complex situations—e.g., outliers, clusters with challenging shapes, non-linearly separable clusters.

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

References

1. Akbilgic, O., Bozdogan, H., Balaban, M.E.: A novel hybrid rbf neural networks model as a forecaster. Stat. Comput. **24**, 365–375 (2014)
2. Andrews, R., Geva, S.: RULEX & CEBP networks as the basis for a rule refinement system. In: Hallam, J. (ed.) Hybrid Problems, Hybrid Solutions, pp. 1–12. IOS Press (1995)
3. Azcarraga, A., Liu, M.D., Setiono, R.: Keyword extraction using backpropagation neural networks and rule extraction. In: The 2012 International Joint Conference on Neural Networks (IJCNN 2012), pp. 1–7. IEEE (2012). https://doi.org/10.1109/IJCNN.2012.6252618
4. Baesens, B., Setiono, R., De Lille, V., Viaene, S., Vanthienen, J.: Building credit-risk evaluation expert systems using neural network rule extraction and decision tables. In: Storey, V.C., Sarkar, S., DeGross, J.I. (eds.) ICIS 2001 Proceedings, pp. 159–168. Association for Information Systems (2001). http://aisel.aisnet.org/icis2001/20
5. Baesens, B., Setiono, R., Mues, C., Vanthienen, J.: Using neural network rule extraction and decision tables for credit-risk evaluation. Manage. Sci. **49**(3), 312–329 (2003). https://doi.org/10.1287/mnsc.49.3.312.12739
6. Barakat, N., Diederich, J.: Eclectic rule-extraction from support vector machines. Int. J. Comput. Inform. Eng. **2**(5), 1672–1675 (2008). https://doi.org/10.5281/zenodo.1055511
7. Benítez, J.M., Castro, J.L., Requena, I.: Are artificial neural networks black boxes? IEEE Trans. Neural Netw. **8**(5), 1156–1164 (1997). https://doi.org/10.1109/72.623216
8. Bologna, G., Pellegrini, C.: Three medical examples in neural network rule extraction. Phys. Medica **13**, 183–187 (1997). https://archive-ouverte.unige.ch/unige:121360
9. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. CRC Press (1984)
10. Calegari, R., Sabbatini, F.: The PSyKE technology for trustworthy artificial intelligence **13796**, 3–16 (2023). https://doi.org/10.1007/978-3-031-27181-6_1, XXI International Conference of the Italian Association for Artificial Intelligence, AIxIA 2022, Udine, Italy, 28 November - 2 December, Proceedings (2022)
11. Castillo, L.A., González Muñoz, A., Pérez, R.: Including a simplicity criterion in the selection of the best rule in a genetic fuzzy learning algorithm. Fuzzy Sets Syst. **120**(2), 309–321 (2001). https://doi.org/10.1016/S0165-0114(99)00095-0
12. Ciatto, G., Calvaresi, D., Schumacher, M.I., Omicini, A.: An abstract framework for agent-based explanations in AI. In: El Fallah Seghrouchni, A., Sukthankar, G., An, B., Yorke-Smith, N. (eds.) 19th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1816–1818. IFAAMAS (May 2020)
13. Craven, M.W., Shavlik, J.W.: Using sampling and queries to extract rules from trained neural networks. In: Machine Learning Proceedings 1994, pp. 37–45. Elsevier (1994). https://doi.org/10.1016/B978-1-55860-335-6.50013-1

14. 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 (Jun 1996)
15. Forina, M., Leardi, R., Armanino, C., Lanteri, S., Conti, P., Princi, P.: Parvus: An extendable package of programs for data exploration, classification and correlation. J. Chemom. **4**(2), 191–193 (1988)
16. Franco, L., Subirats, J.L., Molina, I., Alba, E., Jerez, J.M.: Early breast cancer prognosis prediction and rule extraction using a new constructive neural network algorithm. In: Sandoval, F., Prieto, A., Cabestany, J., Graña, M. (eds.) IWANN 2007. LNCS, vol. 4507, pp. 1004–1011. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73007-1_121
17. 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
18. Hayashi, Y., Setiono, R., Yoshida, K.: A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders. Artif. Intell. Med. **20**(3), 205–216 (2000). https://doi.org/10.1016/s0933-3657(00)00064-6
19. Hofmann, A., Schmitz, C., Sick, B.: Rule extraction from neural networks for intrusion detection in computer networks. In: 2003 IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1259–1265. IEEE (2003). https://doi.org/10.1109/ICSMC.2003.1244584
20. 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
21. 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
22. König, 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
23. Markowska-Kaczmar, U., Trelak, W.: Extraction of fuzzy rules from trained neural network using evolutionary algorithm. In: ESANN 2003, 11th European Symposium on Artificial Neural Networks, Bruges, Belgium, 23–25 April 2003, Proceedings, pp. 149–154 (2003). https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2003-9.pdf
24. Núñez, H., Angulo, C., Català, A.: Rule extraction based on support and prototype vectors. In: Diederich, J. (ed.) Rule Extraction from Support Vector Machines. SCI, vol. 80, pp. 109–134. Springer (2008). https://doi.org/10.1007/978-3-540-75390-2_5
25. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. (JMLR) **12**, 2825–2830 (2011), https://dl.acm.org/doi/10.5555/1953048.2078195
26. Rocha, A., Papa, J.P., Meira, L.A.A.: How far do we get using machine learning black-boxes?. Int. J. Patt. Recogn. Artifi. Intell. **26**(02), 1261001-(1–23) (2012). https://doi.org/10.1142/S0218001412610010
27. 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

28. 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, 31 July - 5 August (2022). https://doi.org/10.24963/kr.2022/57
29. 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, Bologna, Italy, 1–3 Sep, Proceedings, Sun SITE Central Europe, RWTH Aachen University (Oct 2021)
30. 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, vol. 3261, pp. 48–60. Sun SITE Central Europe, RWTH Aachen University (Nov 2022). http://ceur-ws.org/Vol-3261/paper4.pdf
31. 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
32. 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.) EXTRAAMAS 2021. LNCS (LNAI), vol. 12688, pp. 18–38. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-82017-6_2
33. 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, chap. 8, pp. 124–142. Springer (2022). https://doi.org/10.1007/978-3-031-15565-9_8
34. Sabbatini, F., Grimani, C.: Symbolic knowledge extraction from opaque predictors applied to cosmic-ray data gathered with LISA Pathfinder. Aeronau. Aerospace Open Access J. **6**(3), 90–95 (2022). https://doi.org/10.15406/aaoaj.2022.06.00145
35. Saito, K., Nakano, R.: Extracting regression rules from neural networks. Neural Netw. **15**(10), 1279–1288 (2002). https://doi.org/10.1016/S0893-6080(02)00089-8
36. Schmitz, G.P.J., Aldrich, C., Gouws, F.S.: ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. IEEE Trans. Neural Netw. **10**(6), 1392–1401 (1999). https://doi.org/10.1109/72.809084
37. Setiono, R., Baesens, B., Mues, C.: Rule extraction from minimal neural networks for credit card screening. Int. J. Neural Syst. **21**(04), 265–276 (2011). https://doi.org/10.1142/S0129065711002821
38. Setiono, R., Leow, W.K.: FERNN: an algorithm for fast extraction of rules from neural networks. Appl. Intell. **12**(1–2), 15–25 (2000). https://doi.org/10.1023/A:1008307919726
39. Setiono, R., Leow, W.K., Zurada, J.M.: Extraction of rules from artificial neural networks for nonlinear regression. IEEE Trans. Neural Netw. **13**(3), 564–577 (2002). https://doi.org/10.1109/TNN.2002.1000125
40. Setiono, R., Liu, H.: NeuroLinear: from neural networks to oblique decision rules. Neurocomputing **17**(1), 1–24 (1997). https://doi.org/10.1016/S0925-2312(97)00038-6
41. Setiono, R., Thong, J.Y.L.: An approach to generate rules from neural networks for regression problems. Eur. J. Oper. Res. **155**(1), 239–250 (2004). https://doi.org/10.1016/S0377-2217(02)00792-0

42. Steiner, M.T.A., Steiner Neto, P.J., Soma, N.Y., Shimizu, T., Nievola, J.C.: Using neural network rule extraction for credit-risk evaluation. Int. J. Comput. Sci. Netw. Sec. **6**(5A), 6–16 (2006). http://paper.ijcsns.org/07_book/200605/200605A02.pdf
43. Thrun, S.B.: Extracting rules from artifical neural networks with distributed representations. In: Tesauro, G., Touretzky, D.S., Leen, T.K. (eds.) Advances in Neural Information Processing Systems 7, [NIPS Conference, Denver, Colorado, USA, 1994]. pp. 505–512. MIT Press (1994). http://papers.nips.cc/paper/924-extracting-rules-from-artificial-neural-networks-with-distributed-representations